Graph Neural Network Output for Dataset Duplication Detection on Analog Integrated Circuit Recognition System

Arif Abdul Mannan[1], Koichi Tanno[2]
Faculty of Engineering, University of Miyazaki, Miyazaki, Japan[1, 2]
Department of Electrical Engineering, Brawijaya University, Malang, Indonesia[1]

***Abstract*—In the need for artificial intelligence application on the analog circuit design automation, larger and larger datasets containing analog and digital circuit pieces are required to support the analog circuit recognition systems. Since analog circuits with almost similar designs could produce completely different outputs, in case of poor netlist to graph abstraction, larger netlist input circuits could generate larger graph dataset duplications, leading to poor performance of the circuit recognition. In this study, a technique to detect graph dataset duplication on big data applications is introduced by utilizing the output vector representation (OVR) of the untrained Graph Neural Network (GNN). By calculating the multi-dimensional OVR output data into 2-dimentional (2D) representation, even the random weighted untrained GNN outputs are observed to be capable of distinguishing between each graph data inputs, generating different output for different graph input while providing identical output for the same duplicated graph data, and allowing the dataset's duplication detection. The 2D representation is also capable of visualizing the overall datasets, giving a simple overview of the relation of the data within the same and different classes. From the simulation result, despite being affected by the floating-point calculation accuracy and consistency deficiency, the F1 score using floating-point identical comparisons are observed with an average of 96.92% and 93.70% when using CPU and GPU calculations, respectively, while the floating-point rounding calculation is applied. The duplication detection using floating point range comparison is the future work, combined with the study of the 2D GNN output behavior under the ongoing training process.**

***Keywords*—Big data; graph neural network; artificial intelligence; analog circuit design**

I. INTRODUCTION

In computer science, graph have been use in many complex structures, especially structures that focus on the objects (nodes or vertices) and its connections (edges), including chemical structure, human social connection and behavior, electronics circuits, internet World Wide Web, biological structure[1]-[4], etc. In some specific applications, especially in the analog circuit design, the graph can even enable the artificial intelligence (AI) to be applied to the analog electronic design automation (EDA) using graph neural network (GNN) based recognitions [5]-[7].

For global applications, large graph data is something unavoidable. The real-world graph can reach the order of trillion nodes and edges, opening new challenges for graph duplication detection, in case identical graphs exist in the large data[3][4]. In line with the current situation, for large data applications in the analog circuit design field, graph data duplication challenge has also emerged. To generate large datasets, an extraction technique explained in [8] is used. The text-based netlist is used as input, and, split netlist is generated as the output. By using multiple EDA-generated netlists or by using a manually created netlist, duplication of the same circuit extracted from different areas of multiple analog circuits (like a current mirror circuit) are observed to happen frequently [9], as shown in Fig. 1.

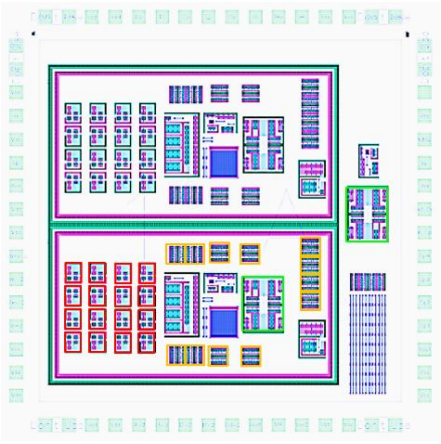

Fig. 1. A dye photograph of analog circuit used as simulation input data in [8] shows multiple uses of the same circuit in various areas inside the red, green, and orange squares

Furthermore, for AI training and circuit recognition, the text-based netlist is converted into graph dataset using netlist to graph abstraction. The recognition accuracy and training performance is dependent on the abstraction technique [10]-[13]. Despite different circuit netlist are used as abstraction process input data, with poor abstraction technique, duplicate graphs occur, reducing the recognition accuracy; thereby, increasing the necessity of graph duplication detection.

The technique to detect graph duplication in large data is already proposed [1][3][4] [14]. However, the technique to detect graph duplication in the literature is directly calculated from the graph data itself. In this study, graph duplication detection technique is proposed by using an integration with GNN recognition, determining the duplication status by calculating the GNN recognition output results.

This study is arranged as follows: The conversion of multi-dimensional data into 2-dimensional (2D) data and the graph duplication detection techniques are introduced in Section II. The accuracy and consistency susceptibility of the floating-point calculation and the methods used to overcome the weakness are described in Section III. The simulation results are provided in Section IV, while the concluding remarks are provided in Section V.

II. GRAPH TO 2-DIMENTION INFORMATION CONVERSION

In this study, a new technique to detect graph duplication on a graph dataset is proposed. The new technique is done by utilizing the GNN output vector representation, calculating the output result and then determining the identical status of the graph. This technique is inspired by an event of a certain failure in the GNN training and recognition of two identical graphs (with different classes) due to the same output and the same error generated. Before discussing the duplication detection technique, the graph datasets and the GNN outputs are first explained.

For graph datasets, considering an owned dataset contains $\mathcal{V}$ number of graph information as $\mathcal{G}(\mathcal{V},\mathcal{E})$ with input features $\{x_v, \forall_v \in \mathcal{V}\}$, has variate number of vertex (node v) and edge (ε) array length on every graph. To detect graph duplication, variate dimension and number of vertices and edges are directly calculated [4]. In this study, to reduce the variation of multiple number of $\mathcal{V}$ and $\mathcal{E}$ array length, GNN output vector representation (OVR) z_v is used as a conversion tool, eliminating the variation of the vertex and edge information into a fixed C dimensional output (C is the number of classes in the dataset). Using this C dimensional data as a source of information, 2D data is then generated.

$$z_v = h_v^{(L)} \quad , \forall v \in \mathcal{V} \tag{1}$$

The GNN OVR z_v is obtained from every neighborhood vector $h_v^{(l)}$ for all $v \in \mathcal{V}$ as shown in Eq. (1). In this study, four GNN models are used, as follows:

1) Graph Convolutional Network (GCN) [19],
2) Graph Isomorphism Network (GIN) [20],
3) Graph Attention Network V2 (GAT) [21], and
4) GraphSAGE (GSG) [22].

The aggregation formula for each GNN used is shown in TABLE I.

TABLE I. GRAPH NEURAL NETWORK FORMULA

Model	Aggregation Formula
GCN	$\boldsymbol{h_v^{(l)} = \sigma\left(b^{(l-1)} + \sum_{k\in N_v} \frac{1}{c_{vk}} W^{(l-1)} h_k^{(l-1)}\right)}$
GIN	$\boldsymbol{h_v^{(l)} = MLP^{(l)}\left((1+\epsilon^{(l)})\cdot h_v^{(l-1)} + \sum_{k\in N_v} h_k^{(l-1)}\right)}$ * *MLP* is a multi-layer perceptron
GSG	$\boldsymbol{h_{N(v)}^{(l)} = mean\left(\{h_k^{(l-1)}, \forall k \in N(v)\}\right)}$ $\boldsymbol{h_v^{(l)} = \sigma\left(W \cdot concat(h_v^{(l-1)}, h_{N_v}^{(l)}, b^{(l-1)})\right)}$ $\boldsymbol{h_v^{(l)} = h_v^{(l)} / \|h_v^{(l)}\|2}$
GATv2	$\boldsymbol{e_{vk}^{(l-1)} = LeakyReLU(a^T \cdot [W^{(l-1)} h_v^{(l-1)} \| W^{(l-1)} h_k^{(l-1)}])}$ $\boldsymbol{a_{vk}^{(l-1)} = softmax_k(e_{vk}^{(l-1)})}$ $\boldsymbol{h_v^{(l)} = \sigma\left(\sum_{k\in N_v} a_{vk}^{(l-1)} W^{(l-1)} h_k^{(l-1)}\right)}$

B. Two Points Distance Value

Multi-dimensional reduction using UMAP and TSNE is already proposed in [15]-[17]; however, since stochastic algorithms are used, the reproducibility is uncertain, especially when using multi-threaded [18]. In this study, a new consistent 2D data generation from C dimension data is proposed. The first dimension of the two data generated is the "distance", simply calculating the distance between two points of data on the C dimensional space into one scalar number. Consider the GNN OVR z_v structure shown in Eq. (2):

$$z_v = \left(z_{v_1}, z_{v_2}, z_{v_3}, \cdots z_{v_C}\right), \forall v \in \mathcal{V} \tag{2}$$

To calculate the "distance" parameter ($prDist$), Pythagorean theorem is used, calculated by choosing one graph OVR result as a reference point, and calculating the $prDist$ of the point under test OVR result by Eq. (3):

$$prDist_p = \begin{cases} \sqrt{\sum_{i=1}^{C} (z_{r_i}^2 - z_{p_i}^2)} & at\ r \neq p \quad r,p \in \mathcal{V} \\ 0 & at\ r = p \quad r,p \in \mathcal{V} \end{cases} \tag{3}$$

With z_r and z_p is vector value of the reference point and point under test in the C dimension output, respectively.

Since the $prDist$ in Eq. (3) uses a square root operation to get the distance, the performance impact is expected if a big dataset is applied as the calculation input. Instead of using the square root, the distance between two points can be calculated using pseudo-distance as shown in Eq. (4). The pseudo-distance is calculated simply by using the sum of the absolute value of every axis's scalar value difference between two points.

$$prpDist_p = \sum_{i=1}^{C} \left|z_{r_i} - z_{p_i}\right| \tag{4}$$

The $prDist$ will have different values from the $prpDist$ with the relationship expressed in Eq. (5). The minimum real distance value of $prDist$ will be equal to the value of $prpDist$ divided by the square root of C in case of equal distance of every axis of the two points take place, as shown in Eq. (6). The maximum distance value of $prDist$ will be equal to the value of $prpDist$ in case of two points axis difference, and has only occurred on one axis x, as shown in Eq. (7), with $1 \leq x \leq C$.

$$\frac{prpDist}{\sqrt{C}} \leq prDist \leq prpDist \tag{5}$$

$$\left|z_{r_1} - z_{p_1}\right| = \left|z_{r_2} - z_{p_2}\right| = \cdots = \left|z_{r_C} - z_{p_C}\right| \tag{6}$$

$$\left|z_{r_i} - z_{p_i}\right| = \begin{cases} prDist_p & when\ i = x \\ 0 & when\ i \neq x \end{cases} \tag{7}$$

The pseudo-distance will have an inaccurate value compared to the real distance. However, the consistency of the calculated value is reliable.

C. Two Points Difference Value

The second data generated after the $prpDist$ data is the "difference", calculating the relative difference between two points of OVR data on the C dimensional space into one scalar number $prDiff$. The relative difference $prDiff$ is also calculated by considering one point of OVR in the C dimensional space as a reference point z_r, and considering another OVR C dimensional point as a point on test z_p.

The first step to calculate the $prDiff$ is by element-wise subtraction between the reference point z_r and the point on test z_p. The element-wise subtraction result is then multiplied by the normalized array of the reference point to produce a relative representation array (RRA). Since the normalized reference point array consists of a fractional number with a range from $0 \leq n_r \leq 1$ and is calculated using Eq. (8), RRA will calculate the element-wise subtraction based on the strong point and weak point of the reference point. The strong points will give large weight to the RRA, and vice versa; weak points will give small weight to the RRA. Therefore, how strong the C dimensional point on a test compared to the reference point is completely described by the RRA.

$$N_r = \frac{z_r - z_{r_{min}}}{z_{r_{max}} - z_{r_{min}}} \tag{8}$$

The final step to calculate the $prDiff$ is by summing all elements on the RRA into a single scalar value. The calculation to generate $prDiff$ value is expressed in Eq. (9).

$$prDiff_p = \sum_{i=1}^{C} (z_{r_i} - z_{p_i}) n_{r_i} \tag{9}$$

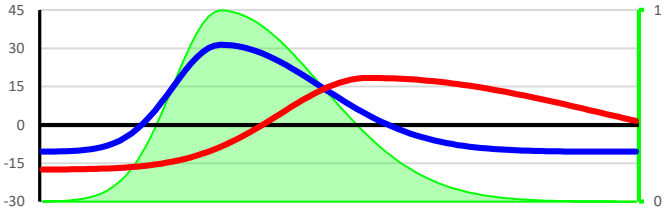

Fig. 2. Positive value of $prDiff$, in area of the normalized reference point (green), the reference point (blue) have higher value relative to the point on test (red).

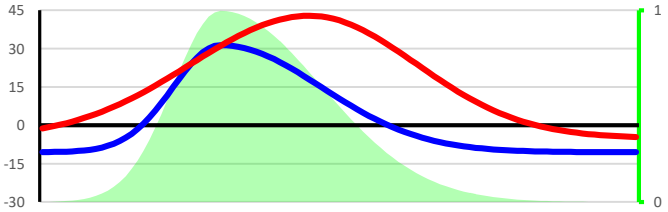

Fig. 3. Negative value of $prDiff$, in area of the normalized reference point (green), the reference point (blue) have weaker value relative to the point on test (red)

Since the relative representation array obtained from z_r subtracted by z_p, the positive value of $prDiff$ represents that the total strong point of the reference point is stronger than point on test as shown in Fig. 2, and the negative value of $prDiff$ representing the total strong point of the reference point is weaker than the point on test as shown in Fig. 3.

D. Duplication Detection

In this study, the duplication detection is done by using identical comparison of the $prpDist$ and $prDiff$ of every graph OVR on the dataset. To begin with, to calculate $prpDist$ and $prDiff$, a reference graph OVR is required, therefore, for convenience, the first graph OVR is selected as reference, specify the index value $r = 0$. With the reference graph OVR has been set, the $prpDist$ and $prDiff$ could be calculated for every other graph OVR in the datasets and the duplication detection can be calculated.

The identical $prpDist$ and $prDiff$ value of two graph under the test have meaning that the "distance" and the "difference" of the two OVR point relative to the reference OVR point is just the same; therefore, the two graphs under the test could be concluded as an identical graph pair. Otherwise, different $prpDist$ and $prDiff$ value of two OVR points means that the two graphs under the test are not identical. The calculation to obtain the graph duplication is shown in Algorithm 1.

Algorithm 1: Duplication Detection

```
PROGRAM Compare()
INPUT z_v
OUTPUT CompOut, CompResult
Compute
for v in V do
prpDist ← cal_dist(z_0, z_v)
prpDiff ← cal_diff(z_0, n_0, z_v)
end for
for i, Dist in prpDist do
CompOut[t][i] ← (prpDist – Dist) == 0
end for
for i, Diff in prpDiff do
CompOut[f][i] ← (prpDiff – Diff) == 0
end for
CompResult ← logical_and(CompOut[t], CompOut[f])
return CompOut, CompResult
```

E. Duplication Detection Scalability

For scalability, in case of a new graph is registered on the dataset, the recalculation of previous $prpDist$ and $prDiff$ are unnecessary. Only new registered graphs are required for $prpDist$ and $prDiff$ calculation, calculated using the same GNN (for obtaining the new registered OVR graph) and by using the same reference OVR graph, same as the other previous already existed graph. The duplication detection for the new graph is then calculated by comparing the new acquired $prpDist$ and $prDiff$ with the previous already calculated $prpDist$ and $prDiff$ one.

III. Accuracy and Consistency

In today's modern computers, floating point numbers represented by the IEEE 754 standard (standard floating-point

or SFP) is used. The floating point calculation in IEEE 754 have limited accuracy and consistency, accurate only about 7 decimal digits for single precision and about 16 decimal digits for double precision, while the bitwise identical output results are not guaranteed even using the identical input and identical mathematical equations [23]-[25].

In this study, SFP is used in all calculations. Starting from the GNN recognition, OVR result calculation, until the identical comparison for the duplication detection. Therefore, some errors due to inaccuracy and inconsistency are to be expected.

A. False Negative Result

As shown in Algorithm 1, the operation to obtain the duplication is by element wise identical comparison. Since the SFP operation is suspected to be affected by the calculation inaccuracy and inconsistency, a test using single graph calculated twice is observed to give a different result, generating a non identical graph detection or false negative result.

In some cases, the result inconsistencies are observed to apply starting from as low as 4 decimal digits of the calculated output (PyTorch tensor calculation using GPU). Therefore, to overcome this inconsistency, rounding algorithms are used with the suspected inaccurate decimal digits that are neglected and replaced with zeros.

In this study, the rounding algorithm is applied at the GNN output vector representation result and at the $prpDist$ and $prDiff$ calculation output. The number of decimal digits maintained is set to 4 decimal digits, for both CPU and GPU calculation. As an example, the value of $x = 1.23456789012 * 10^{-5}$ will be rounded to the value of $x = 1.23450000000 * 10^{-5}$.

B. False Positive Result

The rounding algorithm is expected to reduce the false negative result. However, a new false positive result condition is introduced by applying the rounding algorithm. A pair of the two almost similar graphs, with the GNN OVR differences smaller than the value of the neglected decimals, is observed to be detected as the same identical graph. For the number of decimal digits on the rounding algorithm, as more decimal digits are maintained, the false positive result is expected to appear less, and the false negative result reduction is expected to be weaker.

C. Special False Positive Case

In the duplication detection algorithm, a special false positive case has been observed once. Since the GNN neural network (NN) weight used as the input of the duplication detection is randomly generated and untrained, one occurrence of completely mirrored GNN output is observed as illustrated in Fig. 4.

When the GNN OVR of the reference point (Fig. 4 blue line) has a mirror result characteristic, as expressed in Eq. (10), and when the GNN OVR of graph 1 on test (Fig. 4 green line) and graph 2 on test (Fig. 4 red line) have the same identical OVR results and lie in each mirroring area for the reference point GNN OVR, a special false positive condition is observed. Every element-wise subtraction between the reference and graph 1 on the test completely finds its pair in the subtraction between the reference and graph 2 on a test, as expressed in Eq. (11). Therefore, the total calculated $prpDist$ and $prDiff$ on both graphs will be equal, and the duplication detection will consider graph 1 and graph 2 as identical graph.

$$z_{r_i} = z_{r_j} \quad , i + j = C - 1 \tag{10}$$

$$\left| z_{r_i} - z_{p1_i} \right| = \left| z_{r_j} - z_{p2_j} \right| \tag{11}$$

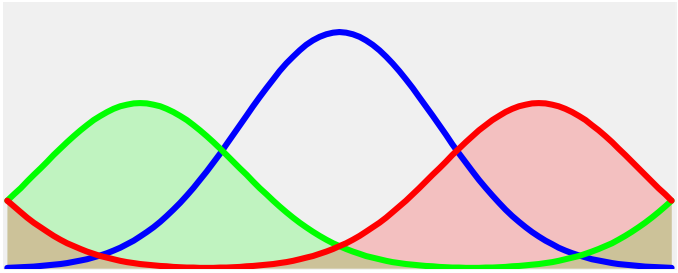

Fig. 4. False positive condition, the reference (blue), test graph 1 (green) and test graph 2 (red). The x-axis is the GNN dimension output from 1 to C, and the y-axis is the scalar value of the GNN output.

D. Second Level Comparison (SLC)

To detect false positive detection cases, a second level comparison (SLC) is proposed. As stated in Section II(D), for the duplication detection, the first graph in the dataset is selected as the reference graph. However, in the SLC, one of two graphs detected as identical graph will be selected as the reference. Therefore, the calculation will only produce one $prpDist$ and one $prDiff$, representing the "distance" value and the "difference" value between the two presumed to be identical graphs under the duplication detection result. If the $prpDist$ and $prDiff$ proved to be equal to 0, the two graphs are indeed identical. The SLC calculation is shown in Algorithm 2.

Algorithm 2: Second Comparison

```
PROGRAM ReComp()
INPUT CompResult, CompResultY, tensorh
OUTPUT CompResult
Compute
for Ident in CompResultY do
DatRef ← tensorh[Ident[0]]
DatNorm ← normtensor(tensorh[Ident[0]])
DatTest ← tensorh[Ident[1]]
rprpDist ← callpdist(DatRef, DatTest)
rprDiff ← callDiff(DatRef, DtaNorm, DatTest)
if rprpDist ≠ 0 or rprDiff ≠ 0 do
CompResult[Ident[0]][Ident[1]] ← False
end if
end for
return CompResult
```

Despite being capable of detecting a false positive case, the SLC is observed to introduce a new false negative case by

detecting the true positive identical pair as a non-identical pair and change into false negative as shown in Fig. 5.

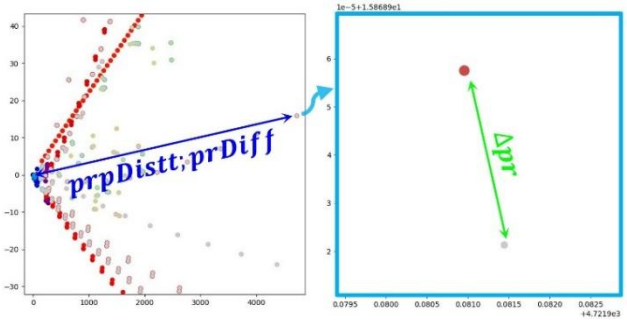

Fig. 5. Second Level Comparison on confirmed true positive identical pair, resulting in a false negative detection due to floating point inaccuracy.

The SFP accuracy is accurate until 7 decimal digits for FP32, as for the value after decimal digit 8 will be considered inaccurate. Since the value of $prpDist$ and $prDiff$ of the graph pair under observation in Fig. 5 is in the order of $5x10^3$, and the delta value between two points in pair Δpr is in the order of 10^{-5}, the Δpr value if these confirmed identical graphs is indeed the result of floating point inaccuracy. In the graph duplication calculation in Algorithm 1, the global reference graph (datasets [0]) is used. Since the Δpr is far from the most significant digit of the $prpDist$ and $prDiff$ (10^{-8} order difference, as a result of the inaccuracy area of SFP calculation), Δpr is observed as 0, and the true positive graph pair is indeed detected as an identical pair. However, in the second level comparison (one of the two graphs becomes the new reference point), the graph duplication detection calculation is based on the Δpr (with value in the order of 10^{-5}) instead of the old $prpDist$ and $prDiff$, which is the subject of inaccuracy. Therefore, the graph pair shown in Fig. 5 (right) is detected as a non-identical pair (false negative) after SLC.

IV. SIMULATION

For dataset duplication detection simulation, the GCN, GIN, GSG, and GAT models are used for generating the GNN OVR before the $prpDist$ and $prDiff$ calculation is done. Furthermore, for the input datasets, the two netlist to graph abstraction technique from [13] is used, with the circuits included in the netlist datasets are obtained using the split technique in [8] as shown in TABLE II.

TABLE II. DATASET FEATURE

Datasets Feature	Quantity
Number of data	2,115
Number of classes	121
Min data per class	1
Max data per class	795
NETLIST length	5,214,505
NETLIST number of line	95,615
Number of MOSFET(s)	74,626
Number of Power Supply(s)	4,234
Number of Resistors(s)	7,804
Number of LC(s)	11

The first two netlists for graph abstraction in [13] are in 1-node with multi-edge and connection node additions (1NMC). Afterwards, the second abstraction is in a 4-node with connection node addition (4NC). In this study, the 1-node with the addition of multi-edge, node connection, and edge direction optimization (1NMC+D) is introduced as the new, more optimized netlist to the graph abstraction technique after the 1NMC duplication detection result is studied. The dataset's confirmed parameters are shown in TABLE III.

TABLE III. DATASETS CONFIRMED PARAMETER

Parameter	1NMC [13]	1NMC + D	4NC [13]
Different Class Dispute	797	1	0
Max Theoretical Accuracy	62.32%	99.95%	100%
Same Class Group Duplication	0	0	0

For all simulations of the duplication detection, including the CPU and GPU calculations, the first graph in the dataset is set as the reference point. The number of decimal digits maintained at the rounding algorithm is also set to 4 decimal digits. A personal computer with 10850K CPU, 64GB of RAM, RTX 4070 Ti GPU, Windows 10 system, and python environment with PyTorch for neural network computing is used for simulation in this research.

A. The 2D Dataset's Visualization

As the $prpDist$ and $prDiff$ of all graphs are already calculated and obtained, a 2D visualization of these two datasets can be achieved. Using the 1NMC and 1NMC+D datasets as an example, the 2D visualization of an untrained random weighted GNN is shown in Fig. 6 and Fig. 7, respectively. The x axis of Fig. 6 and Fig. 7 represents the $prpDist$ value, while the y axis represents the $prDiff$. The most left dark blue dot observed in Fig. 6 (upper) and Fig. 7 (upper) of all GNN outputs is a representation of the reference graph point, representing the $x = 0$ and $y = 0$ of each figure.

As shown in Fig. 6 (upper) and Fig. 7 (upper), the total visualization of all graphs is observed to have different characteristic between each GNN types. It shows that each GNN model indeed has different calculations and output behavior. A particular pattern of the graphs which fall into the same class is also observed.

Fig. 6 (down) and Fig. 7 (down) is the zoomed version of certain areas in Fig. 6 (upper) and Fig. 7 (upper) respectively. Aiming to focus on the visualization of the identical and non-identical graph pairs, Fig. 6 (down), shows the example of "zoomed" multiple two points overlapping each other, indicating multiple two points detected as identical pairs. With the poor netlist to graph abstraction in 1NMC, multiple grey dots are observed to be exactly on top of the red dot (the easiest example to be seen) in all GNN output. On the contrary, as shown in Fig. 7 (down), with optimized direction added to the netlist to graph abstraction technique, the multiple two points overlapping each other are no longer visible. Even in some cases the grey dots are still close to the red dots, as observed in Fig. 7 [down (b)], they are observed to be completely separated.

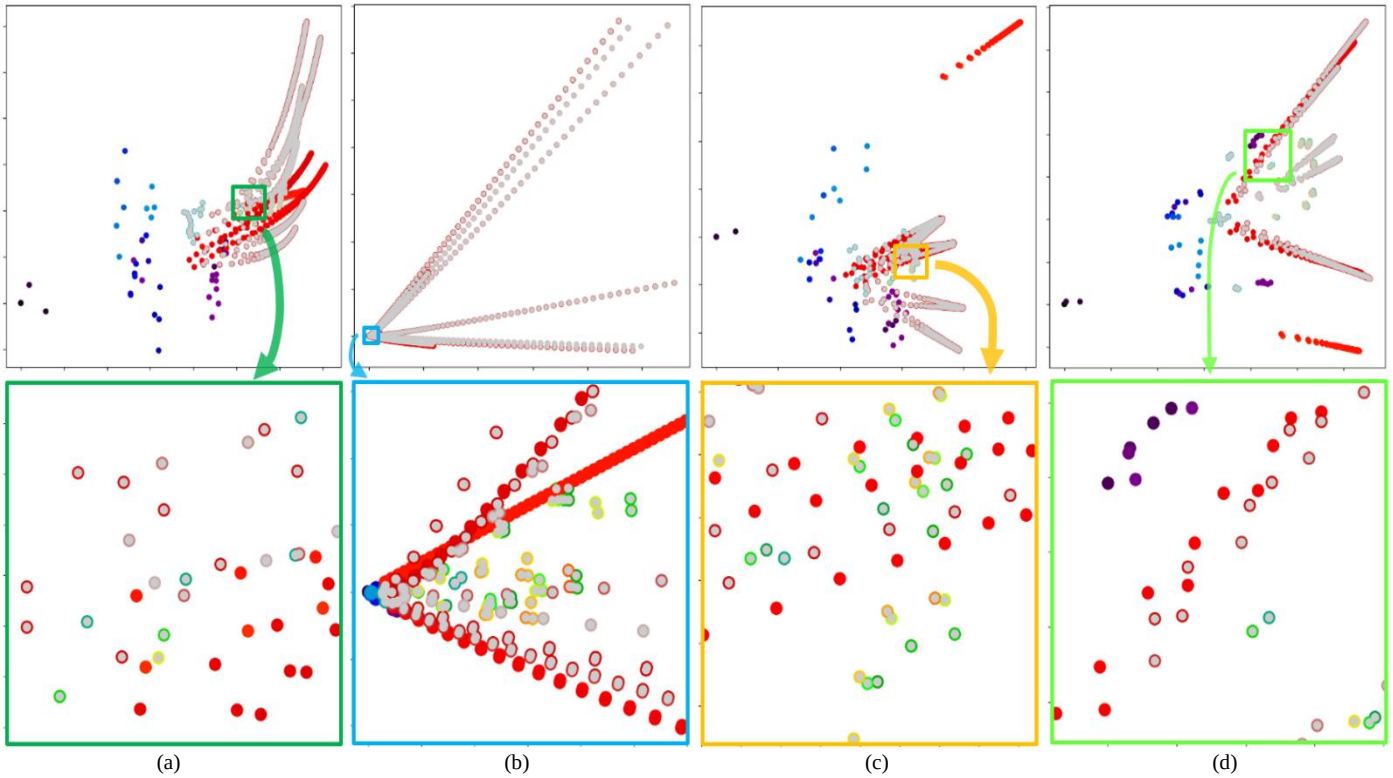

Fig. 6. The 2D output of untrained GCN (a), GIN (b), GSG (c), and GAT (d) of the input dataset using 1NMC netlist to graph abstraction in [13]

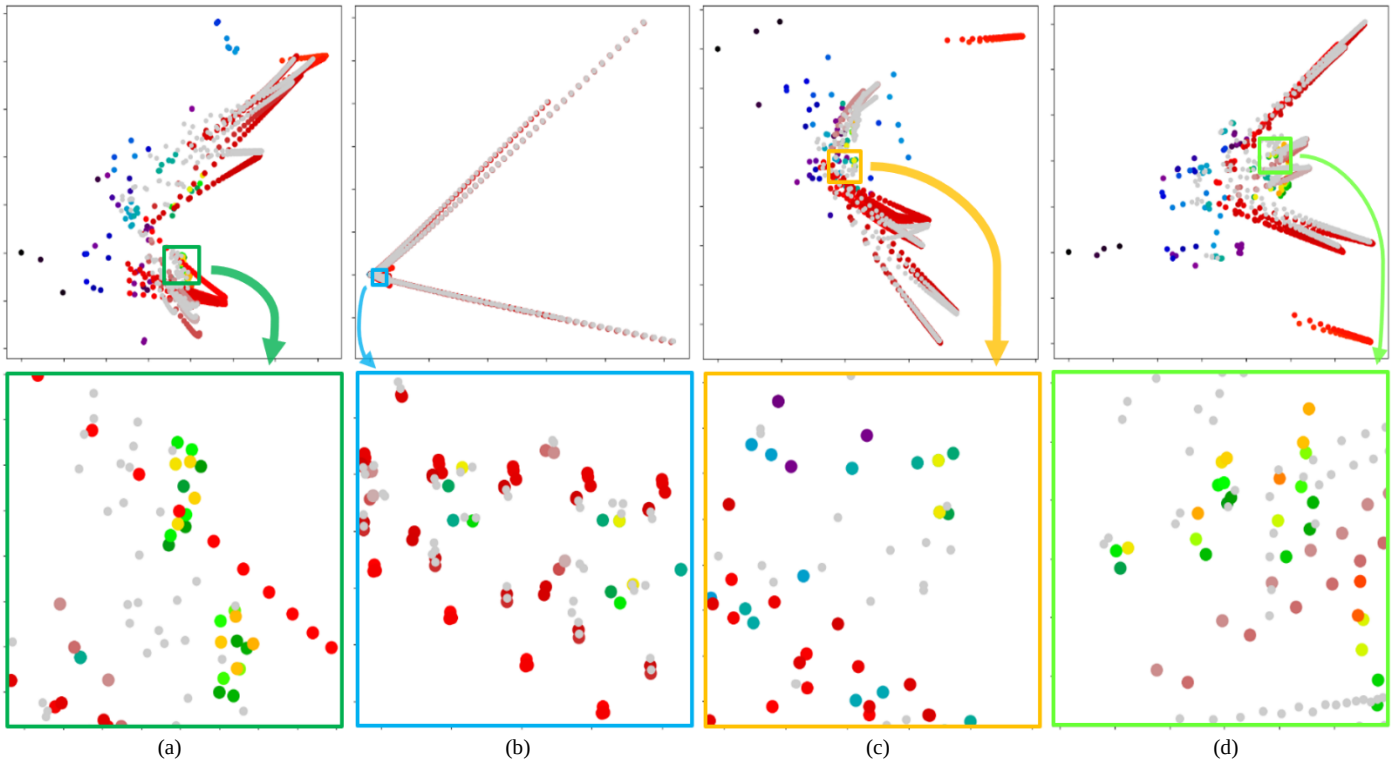

Fig. 7. The 2D output of untrained GCN (a), GIN (b), GSG (c), and GAT (d) of the input dataset using 1NMC + D netlist to graph abstraction

B. *Duplication Detection Result*

The graph dataset duplication detection simulation result for the SFP calculation with combination of SCL and rounded floating point (RFP) calculation with combination of SLC is shown in TABLE I and TABLE V respectively.

TABLE IV. SIMULATION RESULT FOR STANDARD FP CALCULATION COMBINED WITH SCL

Parameter			CPU			GPU		
			1NMC [13]	1NMC + D	4NC [13]	1NMC [13]	1NMC + D	4NC [13]
Standard FP Calculation	Identical Pair Detected	GCN	419	7	1	29	0	0
		GIN	425	7	1	28	0	0
		GSG	383	13	1	5	2	0
		GAT	192	7	1	12	0	0
	Identical Pair False Positive*)	GCN	0	6	1	1	0	0
		GIN	0	6	1	0	2	0
		GSG	1	12	1	0	0	0
		GAT	6	6	1	0	0	0
	Identical Pair False Negative*)	GCN	378	0	0	769	1	0
		GIN	372	0	0	769	1	0
		GSG	415	0	0	792	1	0
		GAT	611	0	0	785	1	0
	Same Class Pair	GCN	0	6	1	1	0	0
		GIN	0	6	1	0	0	0
		GSG	0	11	1	0	1	0
		GAT	6	6	1	0	0	0
	Same Class Pair False Positive*)	GCN	0	6	1	1	0	0
		GIN	0	6	1	0	0	0
		GSG	0	11	1	0	1	0
		GAT	6	6	1	0	0	0
	Detected Dispute Items	GCN	419	1	0	28	0	0
		GIN	425	1	0	28	0	0
		GSG	383	2	0	5	1	0
		GAT	186	1	0	12	0	0
	Calculated training accuracy	GCN	80.19%	99.95%	100%	98.68%	100%	100%
		GIN	79.91%	99.95%	100%	98.68%	100%	100%
		GSG	81.89%	99.91%	100%	99.76%	99.95%	100%
		GAT	91.21%	99.95%	100%	99.43%	100%	100%
	Dispute Items True Positive*)	GCN	419	1	0	28	0	0
		GIN	425	1	0	28	0	0
		GSG	382	1	0	5	0	0
		GAT	186	1	0	12	0	0
FP Calculation + Second level Comparison	Identical Pair Detected	GCN	184	7	0	0	0	0
		GIN	220	7	1	0	0	0
		GSG	326	9	1	0	0	0
		GAT	61	7	1	0	0	0
	Identical Pair False Positive*)	GCN	0	6	0	0	0	0
		GIN	0	6	1	0	0	0
		GSG	1	8	1	0	0	0
		GAT	6	6	1	0	0	0
	Identical Pair False Negative*)	GCN	613	0	0	797	1	0
		GIN	577	0	0	797	1	0
		GSG	472	0	0	797	1	0

		GAT	742	0	0	797	1	0
	Same Class Pair	GCN	0	6	0	0	0	0
		GIN	0	6	1	0	0	0
		GSG	0	8	1	0	0	0
		GAT	6	6	1	0	0	0
	Same Class Pair False Positive*)	GCN	0	6	0	0	0	0
		GIN	0	6	1	0	0	0
		GSG	0	8	1	0	0	0
		GAT	6	6	1	0	0	0
	Detected Dispute Items	GCN	184	1	0	0	0	0
		GIN	220	1	0	0	0	0
		GSG	383	1	0	0	0	0
		GAT	55	1	0	0	0	0
	Calculated training accuracy	GCN	91.30%	99.95%	100%	100%	100%	100%
		GIN	89.60%	99.95%	100%	100%	100%	100%
		GSG	84.59%	99.95%	100%	100%	100%	100%
		GAT	97.40%	99.95%	100%	100%	100%	100%
	Dispute Items True Positive*)	GCN	184	1	0	0	0	0
		GIN	220	1	0	0	0	0
		GSG	382	1	0	0	0	0
		GAT	55	1	0	0	0	0

*) Requires all graph information of the confirmed true positive identical pair, complete with its identical graph counterparts.

TABLE V. SIMULATION RESULT FOR RFP CALCULATION COMBINED WITH SCL

Parameter			**CPU**			**GPU**		
			1NMC [13]	**1NMC + D**	**4NC [13]**	**1NMC [13]**	**1NMC + D**	**4NC [13]**
FP Calculation + Rounding	Identical Pair Detected	GCN	836	54	234	816	54	236
		GIN	839	39	46	800	40	43
		GSG	832	114	39	710	112	37
		GAT	849	25	25	820	24	30
	Identical Pair False Positive*)	GCN	42	53	234	40	53	236
		GIN	45	38	46	45	39	43
		GSG	38	113	39	39	111	37
		GAT	60	24	25	52	24	30
	Identical Pair False Negative*)	GCN	3	0	0	21	0	0
		GIN	3	0	0	42	0	0
		GSG	3	0	0	126	0	0
		GAT	8	0	0	29	1	0
	Same Class Pair	GCN	16	48	81	15	48	81
		GIN	8	22	5	8	22	4
		GSG	15	34	18	15	33	17
		GAT	18	20	5	11	22	8
	Same Class Pair False Positive*)	GCN	16	48	81	15	48	81
		GIN	8	22	5	8	22	4
		GSG	15	34	18	15	33	17
		GAT	18	20	5	11	22	8

	Detected Dispute Items	GCN	805	6	145	787	6	147
		GIN	818	17	40	779	18	39
		GSG	807	78	21	685	78	20
		GAT	881	5	20	796	2	22
	Calculated training accuracy	GCN	61.94%	99.72%	93.14%	62.79%	99.72%	93.05%
		GIN	61.32%	99.20%	98.11%	63.17%	99.15%	98.14%
		GSG	61.84%	96.93%	99.01%	67.61%	96.31%	99.05%
		GAT	61.65%	99.76%	99.05%	62.36%	99.91%	98.96%
	Dispute Items True Positive*)	GCN	794	1	0	776	1	0
		GIN	794	1	0	755	1	0
		GSG	794	1	0	671	1	0
		GAT	789	1	0	768	0	0
FP Calculation + Rounding + Second level Comparison	Identical Pair Detected	GCN	803	14	1	658	8	1
		GIN	782	14	1	404	11	0
		GSG	819	83	1	512	79	0
		GAT	713	14	1	522	10	1
	Identical Pair False Positive*)	GCN	12	13	1	11	7	1
		GIN	9	13	1	2	10	0
		GSG	33	82	1	33	78	0
		GAT	16	13	1	13	10	1
	Identical Pair False Negative*)	GCN	6	0	0	150	0	0
		GIN	24	0	0	395	0	0
		GSG	11	0	0	318	0	0
		GAT	82	0	0	288	1	0
	Same Class Pair	GCN	4	12	1	4	6	1
		GIN	2	12	1	0	9	0
		GSG	12	18	1	12	16	0
		GAT	12	12	1	2	9	1
	Same Class Pair False Positive*)	GCN	4	12	1	4	6	1
		GIN	2	12	1	0	9	0
		GSG	12	18	1	12	16	0
		GAT	12	12	1	2	9	1
	Detected Dispute Items	GCN	797	2	0	653	2	0
		GIN	779	2	0	404	2	0
		GSG	797	65	0	409	63	0
		GAT	719	2	0	519	1	0
	Calculated training accuracy	GCN	62.32%	99.91%	100%	69.13%	99.91%	100%
		GIN	63.17%	99.91%	100%	80.90%	99.91%	100%
		GSG	62.32%	96.93%	100%	76.83%	97.02%	100%
		GAT	66.00%	99.91%	100%	75.46%	99.91%	100%
	Dispute Items True Positive*)	GCN	791	1	0	647	1	0
		GIN	773	1	0	402	1	0
		GSG	786	1	0	479	1	0
		GAT	715	1	0	509	0	0

*) Requires all graph information of the confirmed true positive identical pair, complete with its identical graph counterparts.

The dispute items in TABLE I and TABLE V show the theoretical total number of uncertain graphs in the datasets if the GNN training and recognition are performed. As an example, an identical pair of two graphs has been detected, having indexes 31 and 278 (or expressed as [31, 278]) and having different classes assigned. When the recognition is performed, only one graph will be recognized correctly, and the other graph will be recognized as the other graph class. Therefore, in one identical pair, one graph will be counted as a dispute item.

In case of three identical graphs detected: [24, 334], [24, 1267], and [334, 1267], all with different classes, can also be expressed as [24, 334, 1267], which will have two dispute items. Since only one graph will be recognized correctly, the other two graphs will be detected falsely.

All the results shown in TABLE I and TABLE V are subject to floating-point inconsistency. Therefore, when the simulation is done repeatedly with the same simulation settings and formula, slightly different results are observed.

C. CPU IEEE 754 Calculation + SLC Simulation Result

From using the SFP calculation only, the CPU calculation result for 1NMC datasets is observed to be capable of detecting the graph duplication, with the number of identical graphs detected starting from 192 graphs to 419 graphs detected. However, the number of undetected identical pairs (false negative case) starts from 378 graphs to 611 graphs. According to the false negative result and datasets confirmed parameters in TABLE III, the F1 score started from as low as 37.61% (observed in GAT output result) and as high as 69.56% (observed in GIN), with an average of 60.21% observed.

For calculation after SLC, the number of identical graphs detected decreased, starting from 61 graphs to 184 graphs only, confirming the introduction of false negatives by second level comparison calculation. Therefore, the number of false negative detections observed increased, with the result starting from 472 graphs to 742 graphs. The F1 score is observed to decrease, starting from as low as 12.82% (in GAT) and as high as 57.88% (in GSG), with an average of only 37.87%. There is no change observed in false positive results after SLC calculation.

For calculated theoretical training accuracy from the detected dispute items, the 1NMC dataset shows a large deviation compared to the dataset's parameter shown in TABLE III. The deviation started from 17.78% to 28.89% with an average of 20.98%, and 22.27% to 35.08% with an average of 28.41% are observed in SFP and after SLC calculation result, respectively. Therefore, with the confirmed datasets, a theoretical training accuracy of 62.32%, the deviation of 35.08% (observed in GAT after SLC) is indeed a huge detection error.

For the 1NMC+D datasets simulation result, the SFP calculation could detect "the only one" confirmed identical pairs for all GNN outputs (zero false negatives). However, all GNN outputs are also detecting another non-identical graphs, with an average of 7.5 number of graphs and are reported as identical graphs (false positive case). The false positive cases are observed to belong to the same class, identical pair; therefore, it would not increase the dispute items at all (except 1 false positive dispute item detected in GSG). For calculation after SLC, there is no change observed, except for the GSG false positive reduction from 11 graphs to 8 graphs detected. The GSG dispute items are also reduced from 2 items to only 1 item. For calculated theoretical training accuracy, all GNN shows 0.00% deviation for all SFP calculations and after SLC calculation, except for GSG (SFP result) with 0.04% deviation.

For the 4NC datasets, since it is confirmed zero identical pairs, the false negative detection will always show 0. Therefore, the only parameter that could be considered is false positive detection. From the SFP calculation and after SLC, 1 graph detected as a false positive (same class) is observed in all GNN results except for GNC after SLC (0 false positives). For calculated theoretical training accuracy, all GNN shows 0.00% deviation for all SFP calculations and after SLC.

From the 1NMC, 1NMC+D and 4NC datasets on CPU calculation using SFP and SLC, the duplication detection shows high detection error (up to 93.10% duplication detection incapability error) on the datasets that have many confirmed identical pairs. However, the duplication detection shows high accuracy on datasets with a small, confirmed number of identical pairs. It is concluded that the floating-point inaccuracy and inconsistency are highly impacting the calculation result, resulting two identical graphs calculated with same calculation to give a different result, showing the identical pair as a non-identical pair.

From SFP and SLC calculations, the GAT result is observed showing the worst performance compared to other GNN. There is a possibility observed as the reason GAT output is so inaccurate when using SFP and with SLC calculation. The GAT NN size is so large (stored NN size is 4,455 KB in size) compared to the other GNN NN sizes (215 KB for GCN, 924 KB for GIN, and 351 KB for GSG), indicating more floating-point calculations are performed to obtain the GNN vector representation output. Therefore, the possibility of inaccuracy and inconsistency taking effect is also larger.

D. GPU IEEE 754 Calculation + SLC Simulation Result

The GPU calculation results, by using 1NMC datasets on SFP calculation, were observed to have worse performance compared to the CPU calculation. Due to the number of detected identical pairs being observed only from 5 graphs to 29 graph pairs, the false negative detection number is astonishing, starting from 769 graphs to 792 graphs (confirmed identical pair is 797). Therefore, the F1 score is observed only 1.25% (GSG) to 6.79% (GIN) with an average of 4.45%. For calculation after SLC, the detection capability is even worse. The duplication detection result shows that the calculation is failing to detect any duplication in the datasets (recall score is 0.00%), resulting in the theoretical training accuracy of 100%, although the confirmed theoretical value is 62.32%.

For the datasets 1NMC+D on GPU with SFP and with SLC calculation results, the duplication detection is failing to detect "the only one" confirmed identical pairs for all GNN outputs. The GIN output is even observed on detecting two other graph pairs, with one pair in the same class, raising 1 graph dispute item using the false positive detection. For the detection

capability, the GPU calculation is observed failing to detect the graph duplication (duplication detection capability is 0.00%) on both SFP and SLC calculations.

For GPU calculation on 4NC datasets, the SFP and SLC calculation result shows a consistent outcome in all GNN outputs. The number of identical pairs, false positives, false negatives, and disputed items is zero. The theoretical training accuracy is also observed to be 100% in all cases.

From the 1NMC, 1NMC+D and 4NC datasets on GPU calculation using SFP and SLC, the GPU calculation is concluded to be more severely affected by the floating-point inaccuracy and inconsistency compared to the CPU. The observed output calculation is so affected by this inaccuracy and inconsistency, almost rendering the duplication detection using the proposed method unable to produce the appropriate results at all.

E. CPU Rounding Floating-point + SLC Simulation Result

The CPU calculation results by using 1NMC datasets on RFP calculation, observed to have good performance compared to the SFP result. The number of detected identical pairs observed starts from 832 to 849, with an average of 839 graph pairs. The false negative detection number is observed starting from only 3 graphs until 8 graphs, with an average of 4.25 graph pairs. The F1 score is observed to start from 95.87% (GAT) to 97.48% (GSG), with an average of 96.92%. The number of false positives started from 38 graphs to 60 graphs, with an average of 46.25 graph pairs. For calculation after SLC, the detected false positive and false negative result is observed to have an average of 17.5 and 30.75 graph pairs, respectively. The F1 score is observed to start from 93.43% to 98.88%, with an average of 96.87%. For calculated theoretical training accuracy, the GNN result shows 0.38% to 1.00% and 0.00% to 3.68% deviation for RFP calculation and after SLC calculation, respectively.

For the datasets 1NMC+D on CPU with RFP calculation result, the number of detected identical pairs started from 25 to 114, with an average of 58 graph pairs, with all zeros in all GNN false negative results. The false positive result started from 24 to 113, with an average of 57 graph pairs being observed. For calculation after SLC, the number of detected identical pairs started from 14 to 83, with an average of 31.25 graph pairs, with all zeros in all GNN false negative results. The false positive result is observed to have started from 13 to 82, with an average of 30.25 graph pairs. For calculated theoretical training accuracy, the GNN result shows 0.19% to 3.02% and 0.04% to 3.02% deviation for RFP calculation and after SLC calculation, respectively.

For CPU calculation on 4NC datasets, the RFP calculation result shows a surprising outcome. The number of detected identical pairs increased compared to 1NMC+D, starting from 25 to 234, with an average of 86 graph pairs, with all the detected pairs observed as false positives. For calculation after SLC, the number of detected identical pairs and false positive outcomes is only 1 graph pair in all GNN results. For calculated theoretical training accuracy, the GNN result shows 0.95% to 6.86%, and 0.00% deviation for RFP calculation and after SLC calculation, respectively.

From the 1NMC, 1NMC+D and 4NC datasets on CPU calculation using RFP and SLC, the CPU calculation is capable to produce good results by having very high duplication detection capability. However, despite a high duplication detection capability is observed, the number of false positive detections is rather high, confirming the false negative reduction and new false positive introduction with the rounding calculation application. For SLC calculation, since the false positive results tend to be reduced and the new false negative results tend to be introduced, the detection capability is somewhat observed to be balanced, equalizing between false positive and false negative. This balance result (is it good or bad?) is not discussed in this study.

The RFP output calculation is observed to be less affected by floating-point inaccuracy and inconsistency, rendering the duplication detection using the proposed method able to detect more than 96.14% of the confirmed identical graph pairs.

F. GPU Rounding Floating-point + SLC Simulation Result

The GPU calculation results by using 1NMC datasets on RFP calculation show that the number of detected identical pairs observed started from 710 to 820, with an average of 786.5 graph pairs. The false negative detection number is observed to start from 21 to 126, with an average of 86 graph pairs. The F1 score is observed starting from 89.05% to 96.22% with an average of 93.70%. The number of false positives started from 39 to 52, with an average of 44 graph pairs. For calculation after SLC, the detected false positive and false negative result is observed to have an average of 14.75 and 287.75 graph pairs, respectively. The number of false negative detections observed increased, starting from 150 to 395 graph pairs. The duplication detection capability is observed to decrease, starting from 66.94% to 88.93%, with an average of 76.56%. For calculated theoretical training accuracy, the GNN result shows 0.04% to 5.29% and 6.81% to 18.58% deviation for RFP calculation and after SLC calculation, respectively.

For the datasets 1NMC+D on GPU with RFP calculation result, the number of detected identical pairs started from 24 to 112, with an average of 57.5 graph pair, with all zeros in all GNN (except for GAT, 1 graph pair) false negative results. The false positive result started from 24 to 111, with an average of 56.75 graph pairs being observed. For calculation after SLC, the number of detected identical pairs started from 8 to 79, with an average of 27 graph pairs, with all zeros in all GNN (except for GAT, 1 graph pair) false negative results. The false positive result is observed to start from 7 to 78 with an average of 26.25 graph pairs. For calculated theoretical training accuracy, the GNN result shows 0.04% to 3.64% and 0.04% to 2.93% deviation for RFP calculation and after SLC calculation, respectively.

For GPU calculation on 4NC datasets, the RFP calculation result also shows a surprising outcome. The number of detected identical pairs also increased compared to 1NMC+D, starting from 30 to 236, with an average of 86.5 graph pairs, with all the detected pairs observed as false positives. For calculation after SLC, the number of detected identical pairs and false positive outcomes is only 1 (GCN and GAT) and 0 (GCN and GSG) graph pairs. For calculated theoretical training

accuracy, the GNN result shows 0.95% to 6.95% and 0.00% deviation for RFP calculation and after SLC calculation, respectively.

From the 1NMC, 1NMC+D and 4NC datasets on GPU calculation using RFP and SLC, compared from GPU performance using SFP, the GPU RFP and SLC calculation were observed to be capable of producing good results by having relatively high duplication detection capability. The RFP output calculation is also observed in GPU to be less affected by floating-point inaccuracy and inconsistency, rendering the duplication detection using the proposed method able to detect more than 93.16% and more than 63.90% of the confirmed identical graph pairs in RFP and SLC calculation, respectively.

V. CONCLUSION

In this study, graph duplication detection using the vector representation output of GNN is proposed. The duplication detection starts by recognizing the graph datasets using random weighted untrained GNN and converting the fixed multi-dimensional GNN recognition output into 2D data. The 2D data is later compared to one another to determine if the graph pair under the test is identical or not.

The simulation is also done in this study, using 4 different untrained GNNs, simulated using both CPU and GPU to demonstrate the duplication detection capability despite being affected by floating-point inaccuracy and inconsistency. The best F1 score is obtained by implementing 4-digit decimal rounding floating-point calculation, achieving an average of 96.92% and 93.70% duplication detection from 797 confirmed identical graph pairs using CPU and GPU calculation, respectively, without SLC.

Since the datasets used in this study are generated by using lists of analog circuit modules, the future work in this research is to increase the dataset size with more complex and varied circuits. Other already published graph datasets are also subjected to being tried as input datasets to get a wider application comparison. The optimization of the detection capability by using floating-point range comparison instead of floating-point identical comparison, especially on the second level comparison, along with the behavior of 2D GNN vector output under the training process, is also future work.

ACKNOWLEDGMENT

We would like to express our gratitude to Mrs. Toyama for all the support she provided, including the research environment.

REFERENCES

[1] J. Lee, W.-S. Han, R. Kasperovics, and J.-H. Lee, "An in-depth comparison of subgraph isomorphism algorithms in graph databases," in *Proceedings of the 39th international conference on Very Large Data Bases*. VLDB Endowment, 2012, pp. 133–144.

[2] M. Kraetzl P. Showbridge and D. Ray. Detection of abnormal change in dynamic networks. In *Information, Decision and Control*, 1999.

[3] M. Saltz et all, "Dualiso: An algorithm for subgraph pattern matching on very large labeled graphs," In *IEEE International Congress on Big Data* (BigData Congress), 2014.

[4] A. Mahmood, H. Farooq, J. Ferzund, "Large Scale Graph Matching (LSGM): Techniques, Tools, Applications and Challenges," *International Journal of Advanced Computer Science and Applications* (IJACSA) Vol. 8, No. 4, 2017.

[5] Y Wei, S. Wang, Y. Li, "Graph Theory Based Machine Learning for Analog Circuit Design," *International Conference on Automation and Computing* (ICAC), 2023.

[6] Z. Wu, I. Savidis, "Transfer of Performance Model Across Analog Circuit Topologi with Graph Neural Network," *Workshop on Machine Learning for CAD* (MLCAD), 2022.

[7] S. Sridar, K. Subramanian, "Circuit Recognition Using Netlist," *IEEE Second International Conference on Image Information Processing* (ICIIP), 2013.

[8] A. A. Mannan, K. Tanno, "Netlist Feature Extraction for CMOS Analog Circuit Design Warning System," *ICMLC*, 2024.

[9] Y. Wang, L. Wang, B. Lan, J. Wan, "A Novel Automatic Placement Generation Tool for Current Mirror in Analog Circuits," *2nd International Symposium of Electronics Design Automation* (ISEDA), 2024.

[10] Z. Zheng, X Zhang, Y. Wang, S. He, C. Huang, L. Li, D. Guo, "Classification of Analog Circuit Based on Graph Convolution Network," *International Conference on Anti-counterfeiting, Security, and Identification* (ASID), 2022.

[11] Z. Wu, I. Savidis, "Transfer of Performance Model Across Analog Circuit Topologi with Graph Neural Network," *Workshop on Machine Learning for CAD* (MLCAD), 2022.

[12] S. Sridar, K. Subramanian, "Circuit Recognition Using Netlist," *IEEE Second International Conference on Image Information Processing* (ICIIP), 2013.

[13] K. Hakhamaneshi, M. Nassar, M. Phielipp, P. Abbeel, V. Stojanovic, "Pertaining Graph Neural Network for Few-Shot Analog Circuit Modeling and Design," *IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems*, Vol. 42, NO. 7, JULY 2023.

[14] J. R. Ullmann, "An algorithm for subgraph isomorphism, " *J. ACM*, 23:31–42, January 1976.

[15] M. Mittal, et all, "Dimensionality Reduction Using UMAP and TSNE Technique," *Second International Conference on Advances in Information Technology* (ICAIT), 2024.

[16] Y. Deng, et all, "UMAP for Dimensionality Reduction in Sleep Stage Classification Using EEG Data," *46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* (EMBC), 2024.

[17] E. Myasnikov, "Using UMAP for Dimensionality Reduction of Hyperspectral Data," *International Multi-Conference on Industrial Engineering and Modern Technologies* (FarEastCon), 2020.

[18] "*UMAP Reproducibility*", umap-learn.readthedocs.io. https://umap-learn.readthedocs.io/en/latest/reproducibility.html (accessed April. 24, 2025).

[19] T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks," *5th International Conference on Learning Representations* (ICLR), 2017.

[20] K. Xu, W. hu, J. Leskovec, S. Jegelka, "How Powerful are Graph Neural Networks?," *International Conference on Learning Representations* (ICLR) 2019.

[21] S. Brody, U. Alon, E Yahav, "How Attentive are Graph Attention Networks?," *The Tenth International Conference on Learning Representations* (ICLR), 2022.

[22] W. L. Hamilton, R. Ying, J. Leskovec, "Inductive representation learning on large graphs," *NeurIPS*, 2017, pp. 1025–1035.

[23] "*Numerical accuracy*," pytorch.org. https://pytorch.org/docs/stable/notes/numerical_accuracy.html (accessed April. 14, 2025).

[24] A. Jorgensen, A. Masters, R. Guha, "Assurance of Accuracy in Floating-Point Calculations - A Software Model Study," *International Conference on Computational Science and Computational Intelligence* (CSCI), 2019.

[25] W. Kramer, "A priori worst case error bounds for floating-point computations," *IEEE Transactions on Computers*, 1998.

AUTHORS' PROFILE

Arif Abdul Mannan was born in Malang, Indonesia, on January 23, 1988. Currently, he is a PhD student in University of Miyazaki, Japan. He received S.T degrees from the Faculty of Engineering, Brawijaya University in 2010. He also received master's degrees from Double-Degree Program of Faculty of Engineering in Brawijaya University and Miyazaki University for his M.T and M.E degree in 2013. His research interests are circuit analysis on digital circuits (microprocessor design and application, digital CMOS integrated circuit design, FPGA & VHDL), and analog circuit (analog CMOS integrated circuit design including multi-valued logic). He can be contacted at email: arifabdulmannan@ub.ac.id.

Koichi Tanno was born in Miyazaki, Japan, on April 22, 1967. He received B.E. and M.E. degrees from the Faculty of Engineering, University of Miyazaki, Miyazaki, Japan, in 1990 and 1992, respectively, and Ph. D degree from the Graduate School of Science and Technology, Kumamoto University, Kumamoto, Japan, in 1999. From 1992 to 1993, he joined the Microelectronics Products Development Laboratory, Hitachi, Ltd., Yokohama, Japan. He contributed to the research on low-voltage and low-power equalizers for reading channel LSI of hard disk drives. In 1994, he joined the University of Miyazaki, where he is currently a Professor in the Faculty of Engineering, and a Vice-president (Collaborative Research and Community Cooperation). His main research interests are in analog integrated circuit design, nano-mist sprayers, and its application. Dr. Tanno is a senior member of IEEE. He can be contacted at email: tanno@cc.miyazaki-u.ac.jp.