ARMM: Adaptive Reliability Quantification Model of Microfluidic Designs and Its Graph-Transformer-Based Implementation

Siyuan Liang
The Chinese University of Hong Kong
Shatin, Hong Kong SAR
siyuan.liang@link.cuhk.edu.hk

Meng Lian
Technical University of Munich
Munich, Germany
m.lian@tum.de

Mengchu Li
Technical University of Munich
Munich, Germany
mengchu.li@tum.de

Tsun-Ming Tseng
Technical University of Munich
Munich, Germany
tsun-ming.tseng@tum.de

Ulf Schlichtmann
Technical University of Munich
Munich, Germany
ulf.schlichtmann@tum.de

Tsung-Yi Ho
The Chinese University of Hong Kong
Shatin, Hong Kong SAR
tyho@cse.cuhk.edu.hk

***Abstract*—After decades of development, flow-based microfluidic biochips have become a revolutionary platform for biochemical experiments. To meet the increasingly complex experimental demands, the length and density of channels in these chips grow significantly, which brings about higher defect probabilities. Till now, several methods have been proposed to improve the yield of these increasingly complex chips. However, the effectiveness of these methods cannot be properly evaluated, since there has been no method that systematically analyzes the reliability of a microfluidic design. In this paper, we propose the first mathematical models to quantify the reliability of a microfluidic design by calculating the probability of blockage and leakage defects happening to the design. Besides, we propose a graph-transformer-based method to speed up the calculation, so that designers can have a fast and accurate evaluation of the reliability of a microfluidic design at any scale.**

***Index Terms*—Microfluidic Biochips, Reliability Evaluation, Graph-Transformer**

I. INTRODUCTION

Microfluidic biochips, also known as lab-on-a-chips (LoCs), incorporate many miniaturized devices to carry out complex experiments in molecular biology, biochemistry, and clinical diagnostics [1]–[3]. Compared with traditional experimental platforms, LoCs enjoy many advantages including reactant cost reduction, minimal human intervention, and high-sensitivity [4]. Typically, LoCs consist of two layers: a flow layer containing flow channels and a control layer containing control channels. Reagents and samples are transported among different on-chip devices via flow channels, and the pressure in control channels controls the transportation.

As the complexity of experiments increases, more on-chip devices are required to implement more complex functions, which leads to a significant growth in channel length and density. Consequently, the chance of channel defect increases, and the reliability of LoCs becomes a severe concern [5]–[7]. Due to the unsatisfactory yield, a design may need to be fabricated multiple times to get a well-functioning LoC [8].

To improve the reliability of LoCs, multiple studies have been conducted. Apart from some works seeking a breakthrough in improving material reliability and enhancing manufacturing processes [9], most current works focus on optimizing the design of LoCs. Zhu et al. proposed to construct a logic forest so that certain valves and channel segments can share the same functionalities to realize fault tolerance [10]. Huang et al. proposed to insert backup channels into original designs to provide each device-device connection with several possible paths [11]. Moradi et al. proposed to expand the routing fabric to guarantee the existence of multiple paths through different transposers [12]. Liang et al. proposed a more efficient coding method and merged certain channel segments to reduce the total length and density of channels in multiplexers [13].

In general, current studies focus on exploring structural variants of the design to improve the reliability. However, there is yet no method to evaluate the reliability of a LOC design. Thus, the degree of the improvement cannot be quantified, which makes it challenging for designers to effectively improve their designs.

In this paper, we propose the first quantification model that is able to fit in different production situations and provide objective and quantitative measurements of the reliabilities of both the flow and the control layers of LoC designs. Furthermore, we propose a graph-transformer-based implementation, which learns the outputs of the proposed quantification models, and gives fast and accurate predictions for LoC designs of different scales.

The rest of the paper is organized as follows: Section II introduces common defects in LoCs and the graph transformer. Section III presents details of the proposed reliability quantification models and how to adapt them to different manufacturing processes. Section IV shows the graph representation and feature extraction, the network architecture, the training dataset generation, and the overall flow of the graph-transformer-based implementation. Section V illustrates experimental results, and Section VI draws the conclusion.

II. PRELIMINARIES

A. Defects in Microfluidic Biochips

Recent advances in fabrication, such as polydimethylsiloxane (PDMS) and dense integration of microvalves, have enabled the rapid development of microfluidic biochips [14]. However, due to the limitation of material and processing precision, as the total length and density of channels increase, defects happen more frequently to microfluidic channels [5].

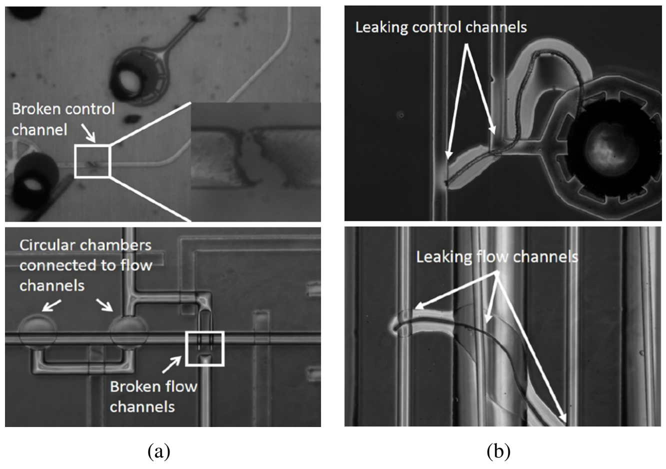


Fig. 1: Channel defects [6]. (a) Blockage. (b) Leakage.

There are two common types of channel defects [6]:

1) Blockage: accidentally broken channel segments. Fig. 1(a) shows examples of broken control channels and flow channels. A blockage is usually caused by environmental particles and imperfect silicon wafer mold [5]. Besides, reagents or cells may also get clogged in flow channels and block the fluid transportation. For flow channels, blockage prevents reagents from reaching the reactor. For control channels, blockage prevents pressure from reaching the valves, leading to losing control over the valves along the channel.

2) Leakage: unexpected connections between channels. Fig. 1(b) shows examples of leaking control channels and flow channels, respectively. For flow channels, reagents may leak from one channel to the other, resulting in dilution and contamination. For control channels, pressure may leak from one channel to the other, resulting in the unanticipated closure of valves.

According to previous studies [5], [6], blockage happens to different parts of a design nearly randomly, while the probability of leakage increases as the length of parallel channels increases or the distance between two neighboring channels decreases. The probabilities of blockage and leakage defects are also closely related to the chosen technology in the manufacturing process.

B. Graph Transformer

First proposed in 2017, the transformer has achieved huge success in the field of Natural Language Processing (NLP) [15]. Since then, numerous follow-up works have been conducted to adapt the transformer to other fields. Among the many adaptations for graph classification and regression, Universal Graph Transformer Variant 2 (UGformer V2) is one of the latest works and has outstanding performance [16]–[18].

As shown in Fig. 2, the UGformer V2 layer first applies a transformer self-attention network to learn the local features of all nodes. The following Graph Convolutional Networks (GCNs) are then utilized to incorporate the structural information of graph $\mathcal{G}$ into learning, and combine the representations obtained before to get the output graph embedding. Formally,

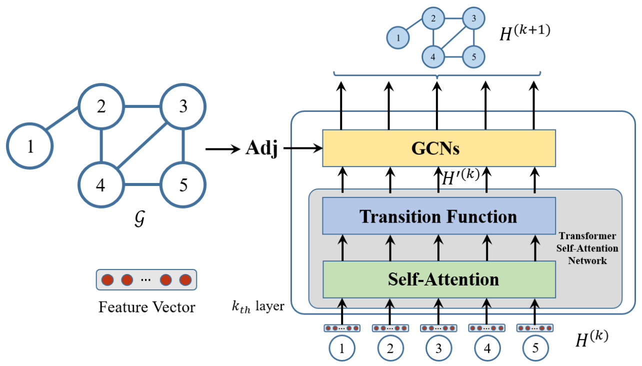


Fig. 2: Illustration of UGformer V2 [16].

we can define a UGformer V2 layer as:

$$\begin{aligned} H'^{(k)} &= ATT(H^{(k)}Q^{(k)}, H^{(k)}K^{(k)}, H^{(k)}V^{(k)}), \\ H^{(k+1)} &= GCN(Adj, H'^{(k)}), \end{aligned} \tag{1}$$

where $H'^{(k)}$ stands for the output of the transformer self-attention network, $H^{(k)}$ is the output of the k_{th} UGformer V2 layer, $H^{(0)}$ is the feature matrix of all nodes in $\mathcal{G}$, Adj is the adjacency matrix of $\mathcal{G}$, ATT represents the transformer self-attention network, and GCN represents GCNs. $Q^{(k)}$, $K^{(k)}$, and V^k are query-projection, key-projection, and value-projection matrices, respectively.

III. QUANTIFICATION MODELS OF DEFECTS PROBABILITIES

The proposed quantification models are applicable to both flow and control layers. We decompose LoC designs into grids with adjustable granularity to ensure the generality of the proposed models [10], [19]. Fig. 3 gives examples of grid-formed designs of control and flow layers.

A. Model of Blockage Probability

Blockage happens to different parts of a design nearly randomly [6]. Hence, when a segment fails, the chance that its backup segments can take over the original fluid transportation is the most important factor in determining the reliability against blockage. In this case, we propose an adaptive propagation-based quantification model to calculate

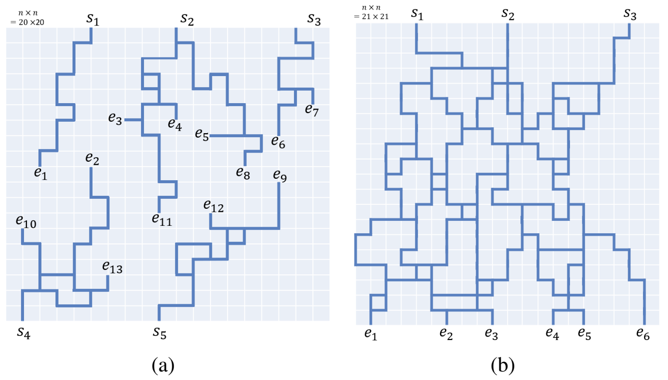

$n \times n = 20 \times 20$
s_1
s_2
s_3
e_1 e_2 e_3 e_4 e_5 e_6 e_7 e_8 e_9 e_{10} e_{11} e_{12} e_{13}
s_4
s_5
(a)
$n \times n = 21 \times 21$
s_1
s_2
s_3
e_1 e_2 e_3 e_4 e_5 e_6
(b)

Fig. 3: Examples of randomly generated grid-formed designs (a) Randomly generated control layer design. (b) Randomly generated flow layer design.

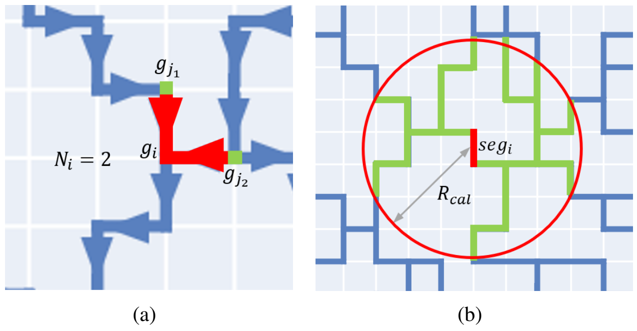


Fig. 4: Schematics of defects probability quantification models. (a) Schematic of the blockage probability quantification for an example g_i with $N_i = 2$ input segments. (b) Schematic of the leakage probability quantification model.

$P_{blockage}$, which is the probability of the design failing owing to blockage.

The fluid transportation in channels of LoCs is similar to the flow of current in electronic circuits [20]. By replacing the current flow I_k with the transportation of fluid T_k, the potential difference ΔU_k with the pressure difference ΔP_k, and the resistance R_k with the channel length L_k, we can adopt Kirchhoff's first and second laws [21] to derive directions of fluid transportation in different designs:

$$\begin{aligned} \textstyle\sum_{k=1}^{n} I_k = 0 &\quad\Longrightarrow\quad \textstyle\sum_{k=1}^{n} T_k = 0 \\ \textstyle\sum_{k=1}^{m} \Delta U_k = 0 &\quad\Longrightarrow\quad \textstyle\sum_{k=1}^{m} \Delta P_k = 0 \\ \Delta U_k = R_k \times I_k &\quad\Longrightarrow\quad \Delta P_k = L_k \times T_k \end{aligned} \tag{2}$$

where n stands for the number of branches with current/ fluid flowing toward or away from the node, and m is the number of components in a closed loop.

Arrows in Fig. 4 (a) illustrate the calculated fluid transportation directions of part of the example design shown in Fig. 3 (b). Suppose that grid point g_i has N_i input segments, and g_j is the neighboring grid point along the input segment of grid point g_i, as shown in Fig. 4 (a). Then $p_{blockage_i}$, the probability that fluid from the starting points fails to reach g_i, can be derived by the following formula:

$$p_{blockage_i} = \prod_{j=1}^{N_i} pb_{i,j}$$
$$pb_{i,j} = p_{blockage_j} + (1 - p_{blockage_j}) \times (1 - (1 - pb_{unit})^{l_{i,j}}) \tag{3}$$

where pb_{unit} stands for the probability of each segment with unit length being broken, and $pb_{i,j}$ represents the probability of the fluid failing to reach g_i regarding g_j. Such failures can be categorized into two types: i) the fluid from the starting points fails to reach g_j; ii) the fluid from the starting points succeeds in reaching g_j, but the connection between g_i and g_j is broken. Further, $l_{i,j}$ stands for the length of the connection between g_i and g_j, here it is equal to d_{unit}, the chosen granularity. As shown in Fig. 5(a), $pb_{i,j}$ will increase with $l_{i,j}$, which is consistent with the law summarized in [6].

Since the quantification model focuses on the robustness of LoC designs, we assume that the external fluid and pressure

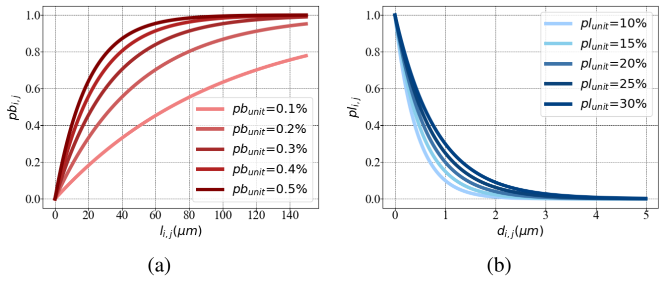


Fig. 5: Adaptive Relationships between $pb_{i,j}$ and $l_{i,j}$, $pl_{i,j}$ and $d_{i,j}$. (a) Different relationships between $pb_{i,j}$ and $l_{i,j}$ by changing pb_{unit}. (b) Different relationships between $pl_{i,j}$ and $d_{i,j}$ by changing pl_{unit}.

supply is 100% reliable. In this case, we set all $p_{blockage_{s_i}}$ to 0, where s_i are indices of the starting points. By iteratively applying Eq. 3, we propagate our calculation from the starting points to the ending points, and eventually get the probabilities that fluid fails to reach the N_e ends, marked by $p_{blockage_{e_i}}$, where e_i are indices of the ending points. The design is regarded as well-functioning only when fluid can reach all the ends. Thus, the probability of the design failing owing to blockage, denoted as $P_{blockage}$, can be calculated by:

$$P_{blockage} = 1 - \prod_{l=1}^{N_e} (1 - p_{blockage_{e_i}}) \tag{4}$$

B. *Model of Leakage Probability*

The probability of leakage increases as the length of parallel channels increases or the spacing between channels decreases [6]. Hence, the distance between a segment and its nearby segments is the key factor in determining the reliability against leakage. In this case, we propose an adaptive neighboring-region-scanning-based quantification model to calculate $P_{leakage}$, which is the probability of the design failing owing to leakage.

As shown in Fig. 4(b), the position of a segment is represented by the Cartesian coordinates of its middle point. Taking segment seg_i marked by the red line as the center, we select all other segments seg_j within the red circle of radius R_{cal}, which is a manually-set threshold, and mark them green. Then we calculate the probability of leakage happening between seg_i and these selected segments, and combine them to derive $p_{leakage_i}$, the overall probability of leakage happening to seg_i. Notably, the larger R_{cal} is, the more accurate $p_{leakage_i}$ we will have. Suppose that a total of N_s segments are selected, then:

$$p_{leakage_i} = 1 - \prod_{j=1}^{N_s} (1 - pl_{i,j})$$
$$pl_{i,j} = {pl_{unit}}^{d_{i,j}} \tag{5}$$

where pl_{unit} represents the leakage probability of two segments with unit distance, $pl_{i,j}$ is the probability of leakage happening between seg_i and seg_j, which is exponentially related to $d_{i,j}$, i.e. the Euclidean distance between seg_i and seg_j. As shown in Fig. 5(b), $pl_{i,j}$ will decrease with the

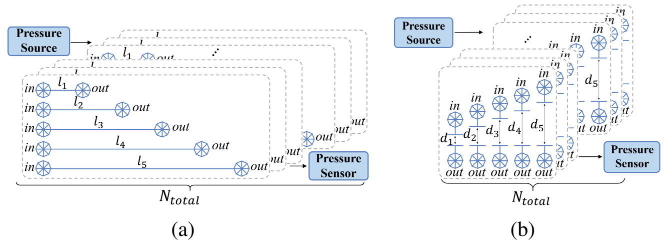


Fig. 6: Schematics of collecting statistical data for (a) blockage and (b) leakage.

increase of $d_{i,j}$, which is consistent with the law summarized in [6].

As a matter of fact, when leakage happens between channels that have exactly the same sources and targets, i.e. a channel and its backup channel, the leakage will not affect the normal operation of the chip. This feature can be taken into consideration by explicitly defining the backup channels and modifying Eq. 5 to only calculate $pl_{i,j}$ for all seg_j that are not among backup channels of seg_i. Under this setting, any potential leakages considered will result in inappropriate transportation of fluids. In this case, $P_{leakage}$ is equal to the probability of the design having at least one leakage. Assume there are N_{seg} segments in the given design, and d_{unit} is the chosen granularity, $P_{leakage}$ can then be derived by Eq. 6. If we keep the same R_{cal} and change the granularity, the number of selected segments will change quadratically, thus distort the result. Hence, the item ${d_{unit}}^2$ is used in the formula to correct the error caused by the change of granularity.

$$P_{leakage} = 1 - \left(\prod_{i=1}^{N_{seg}} (1 - p_{leakage_i})\right)^{{d_{unit}}^2} \tag{6}$$

C. Adaptation to Different Manufacturing Processes

In order to quantify the reliabilities of designs under a specific manufacturing process, it is necessary to choose suitable pb_{unit} and pl_{unit} that can best reflect the precision of the process. As shown in Fig 5(a) and (b), the choice of pb_{unit} and pl_{unit} influences how the probabilities of blockage $pb_{i,j}$ and leakage $pl_{i,j}$ are affected by the length of a channel $l_{i,j}$ and the distance between two unit-length-long channels $d_{i,j}$, respectively. Here, we give a detailed 2-stage guidance of how to adapt the proposed quantification models to a specific manufacturing process.

1) Stage I: At this stage, one should collect statistical data of defects happening under different design conditions during manufacturing. Since real designs have very different and complex structures, statistical data collected from them are affected by too many factors and lack of consistency, which makes the data unable to properly reflect the pattern of reliability of the manufacturing process. Hence, a systematic approach is proposed to manufacture test designs and generate statistical data in the following.

As shown in Fig 6, we propose to manufacture channels with n different lengths $l_1, l_2, ..., l_n$ as the blockage testing set, and channel pairs of unit length but with n different distances in between $d_1, d_2, ..., d_n$ as the leakage testing set. Each set is repeated for N_{total} times to form the blockage and leakage testing groups. The larger N_{total} is, the more trustworthy the statistics will be. Then we can apply high pressure to in ports and detect the pressure level at out ports. In the blockage testing group, if the detected pressure is low, there is a blockage in the channel. In the leakage testing group, if the detected pressure is high, there is a leakage between the channel pair. For each l_s and d_s, we count the number of blocked channels, denoted as $N_b(l_s)$, and the number of leaking channel pairs, denoted as $N_l(d_s)$, then we can derive the statistical data by:

$$pb_{statistic}(l_s) = \frac{N_b(l_s)}{N_{total}}, \; pl_{statistic}(d_s) = \frac{N_l(d_s)}{N_{total}} \tag{7}$$

2) Stage II: At this stage, we select suitable pb_{unit} and pl_{unit}, whose corresponding curves can best fit the collected statistical data.

Since conducting infinite tests is impractical, statistical data will inevitably have deviations. Let $b_{dev}(l_s)$ and $l_{dev}(d_s)$ be deviations of $pb_{statistic}(l_s)$ and $pl_{statistic}(d_s)$ respectively, and they are calculated by:

$$\begin{aligned} b_{dev}(l_s) &= |pb_{i,j}(l_s) - pb_{statistic}(l_s)| \\ l_{dev}(d_s) &= |pl_{i,j}(d_s) - pl_{statistic}(d_s)| \end{aligned} \tag{8}$$

The corresponding expectations can be calculated as:

$$\begin{aligned} E(b_{dev}(l_s)) &= \sum_{q=0}^{N_{total}} P_{N_b(l_s)}(q) \times |pb_{i,j}(l_s) - \frac{q}{N_{total}}| \\ E(l_{dev}(d_s)) &= \sum_{q=0}^{N_{total}} P_{N_l(d_s)}(q) \times |pl_{i,j}(d_s) - \frac{q}{N_{total}}| \end{aligned} \tag{9}$$

where $P_{N_b(l_s)}(q)$ and $P_{N_l(d_s)}(q)$ represent the probabilities of $N_b(l_s) = q$ and $N_l(d_s) = q$, respectively, and they can be expressed as:

$$\begin{aligned} P_{N_b(l_s)}(q) &= C_q^{N_{total}} \times pb_{i,j}(l_s)^q \times (1 - pb_{i,j}(l_s))^{N_{total}-q} \\ P_{N_l(d_s)}(q) &= C_q^{N_{total}} \times pl_{i,j}(d_s)^q \times (1 - pl_{i,j}(d_s))^{N_{total}-q} \end{aligned} \tag{10}$$

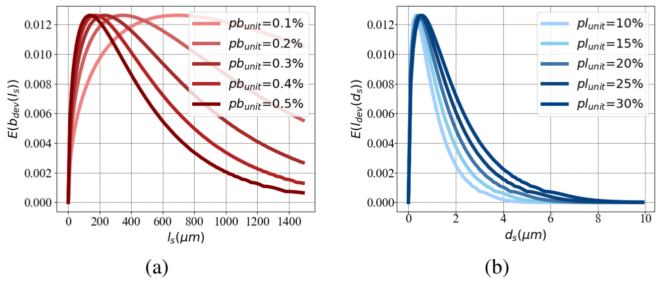


Fig. 7: Expectations of deviations. (a) Different relationships between $E(b_{dev}(l_s))$ and l_s by varying pb_{unit}. (b) Different relationships between $E(l_{dev}(d_s))$ and d_s by changing pl_{unit}.

Fig. 7 shows expectations of $b_{dev}(l_s)$ and $l_{dev}(d_s)$ with the same settings of pb_{unit} and pl_{unit} in Fig. 5, respectively. According to the figures, though curves have different shapes, they share the same trend that expectations of deviations become much smaller when l_s and d_s are larger.

To perform curve fitting, which involves selecting appropriate pb_{unit} and pl_{unit} values that produce curves most closely resembling the statistical data for the relationships between $pb_{i,j}$ and $l_{i,j}$, and $pl_{i,j}$ and $d_{i,j}$, the most straightforward and intuitive approach is to minimize the mean absolute error (MAE). However, MAE simply adds the deviations up, and ignores the different significance of these deviations, which limits its performance. In this case, we propose to minimize a weighted MAE, which can achieve better performance:

$$\begin{gathered} Min\ \sum_{s=1}^{n} w_s \times |pb_{i,j}(l_s) - pb_{statistic}(l_s)| \\ Min\ \sum_{s=1}^{n} w_s \times |pl_{i,j}(d_s) - pl_{statistic}(d_s)| \\ \sum_{s=1}^{n} w_s = 1 \end{gathered} \tag{11}$$

where $w_1, w_2, ..., w_n$ follows an increasing order to ensure the fitting results refer more to the statistical data collected with larger l_s and d_s, which have smaller expected deviations according to Fig. 7.

IV. GRAPH-TRANSFORMER-BASED IMPLEMENTATION

The proposed quantification models have a trade-off between accuracy and time usage. If using a fine enough granularity d_{unit} or a very large R_{cal} to seek the highest accuracy, the time usage will increase significantly, making it impractical to use the models as feedback to automatically optimize original designs. Hence, we propose a machine-learning-based implementation that slightly sacrifices the accuracy but can significantly reduce the time usage.

Since the proposed implementation must face designs of very different scales, the scalability of the chosen machine-learning method is of great importance. Considering this, we represent designs of different scales as graphs with different numbers of vertices and weights of edges, and utilize the graph transformer, which is among the currently most performant graph learning methods, to learn from the proposed quantification models and achieve fast and accurate predictions.

A. Graph Representation and Feature Extraction

Different grid-formed LoC designs are the inputs of our method. To start with, we need to transform the original grid-formed designs into graphs so that the data can be fed into the graph transformer.

For a given grid-formed LoC design, simply regarding all grid points as nodes will get the critical information lost in a sea of useless information, which will only burden our training and will not lead to better training results. Hence, we propose to extract only the key grid points crucial to the design, and then form nodes accordingly.

There are four kinds of points that characterize an LoC design: starting points, ending points, corners, and interchanges, i.e. grid points that are connected with more than two channel segments. The discarded grid points are empty spaces and interior points of straight channel segments, which do not contribute much to characterize the structure of a design.

For each node, we use a 6-dimensional vector to represent its features. The first two elements represent the location of the node, and the remaining four elements are one-hot elements representing the kind of the point. As for the edges of a graph, we assign the Euclidean distances between nodes to corresponding edges. Since grid points not extracted to form nodes are either empty spaces or interior points of straight channel segments, we can directly use $d_{i,j}$, as mentioned in the previous section, to represent the Euclidean distances between nodes. For nodes that are not connected with each other, $d_{i,j}$ is set to infinity to denote the disconnection.

TABLE I: The Graph Transformer Configuration

Layer	Type	Input-Dim	Output-Dim	Activation
1	Fully-connected	6	256	N/A
2-4	UGformer V2	256	256	N/A
5	Soft attention	256	256	Relu
6	Sum & max pooling	256	256	N/A
7	Fully-connected	256	1	N/A

B. Network Architecture

The principles of the proposed quantification models of blockage and leakage probabilities are different. To ensure a better training effect, we train two networks to predict reliabilities against leakage and blockage respectively. Each network takes the graph with node features and weighted edges as the input, and outputs a 1-dimensional vector, i.e. a value, as the prediction.

As shown in TABLE I, the structure of the network consists of five parts: i) the pre-encoder, which linearly encodes the input and does dropout to alleviate overfitting; ii) the UGformer V2 [16], which further extracts features from the graph; iii) the self-attention block, which helps to learn the latent connections between distant nodes; iv) the sum & max pooling, which can take into account both the graph structure and key nodes' features to get better embeddings [22]; v) the predicting block, which consists of a fully-connected layer, and gives a prediction vector according to the processed graph embeddings. Dropout is also applied in this block to alleviate overfitting.

Among the many graph transformers [16]–[18], we choose UGformer V2 as the core part of our network, because of its easy training and efficient functions [16]. Each UGformer V2 layer takes the current graph embedding and the adjacency matrix as the input and produces the updated graph embedding. To learn potential patterns of graphs, we use a 6-256 fully-connected layer with dropout to extract features of nodes with overfitting prevention, and utilize three 256-256 cascaded UGformer V2 layers to update graph embeddings.

Soft attention is adopted to strengthen the understanding of nodes [23]. The graph embedding goes through a 256-256 fully-connected layer with the activation of $Relu$ and turns into the intermediate results. The same graph embedding also goes through the other 256-1 fully connected layer with the $Sigmoid$ activation to get each node's attention weights. The updated graph embedding is derived by multiplying the intermediate results and attention weights.

Then, we use sum & max to further strengthen the understanding of the graph structure and key nodes' features [22]. In the end, a 256-1 fully-connected layer is used to derive the 1-dimensional prediction vector from the last graph embedding.

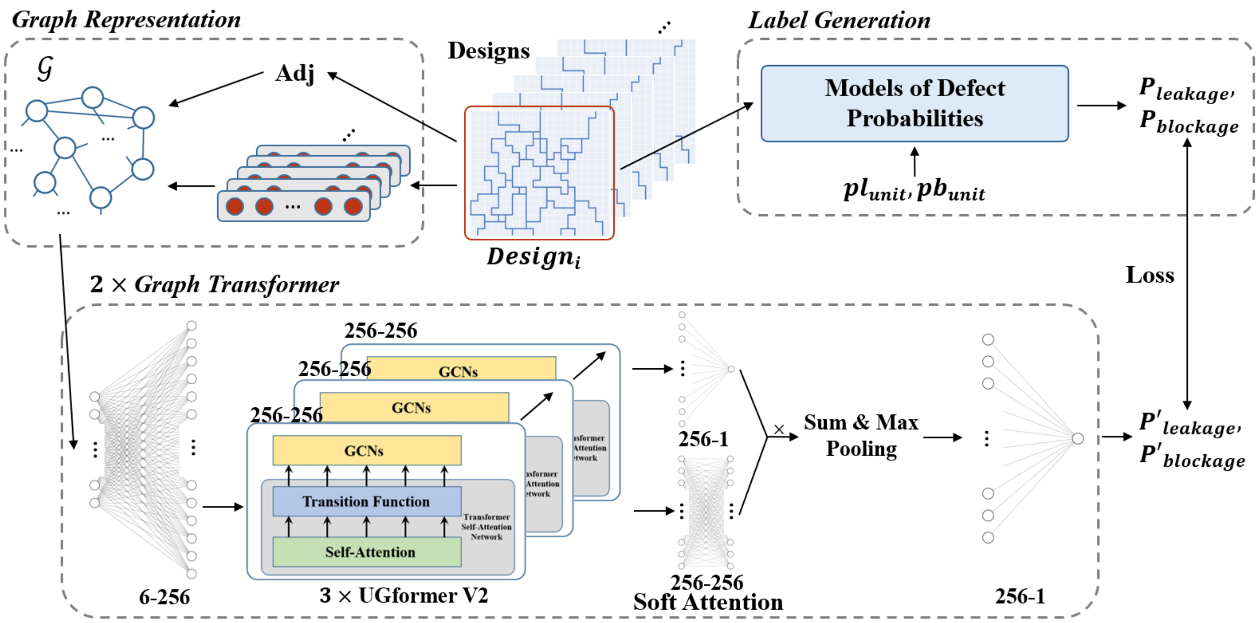


Fig. 8: The graph transformer training flow.

C. Training Dataset Generation

For learning-based methods, the training dataset is of great importance. Even with a highly efficient learning model, there is a substantial possibility that the learning performance will not be satisfactory if the dataset is insufficiently large or the data quality is subpar.

In order to enhance our model's ability to infer and handle diverse designs, an optimal training dataset should consist of a substantial number of LoC designs that exhibit notable structural variations and do not conform to a uniform design pattern. Considering that it is difficult to obtain such dataset through real designs, we design a four-stage randomized design generation algorithm to generate the training dataset:

- Randomly generate the dimensions $n \times n$ of a design, and then a corresponding empty grid will be produced.
- Randomly generate the number of starting and ending points. Since the trained graph transformer is expected to be applicable to both control and flow layers, we randomly decide whether to generate a control layer or a flow layer design. If generating a control layer design, starting points will be randomly placed on the top and bottom of the design, and ending points will be randomly placed inside the design. Otherwise, starting points will be randomly placed on the top, and ending points will be randomly placed on the bottom of the design.
- Randomly generate channel segments in the design. Once the generation is finished, check if paths from starting points can reach all ending points. If not, the design will be regarded as invalid, and this stage will be re-executed.
- Apply the proposed quantification models of defects probabilities on the design, and add the derived values into a table recording the sequence number of designs and their corresponding defects probabilities.

By repeatedly executing the algorithm, we generate 26411 random LoC designs and a table storing the blockage and leakage probabilities of these designs, which are regarded as the ground truth in the training.

D. Overall Flow

Fig. 8 shows the overall flow of the graph transformer training, which consists of three stages: graph representation, label generation, and graph transformer training.

As illustrated in Section IV A, each grid-based design is examined at the beginning of the graph representation stage, and key grid points like starts, ends, corners, and interchanges are selected as nodes of the graph. Then their corresponding feature vectors and the adjacency matrix are formed. Combining them, the corresponding graph representation $\mathcal{G}$ is derived.

In the label generation stage, quantification models of defects probabilities are adjusted by following the guidance proposed in Section III C. Then the adjusted quantification models are used to calculate $P_{leakage}$ and $P_{blockage}$ of different designs generated in Section IV C, and these probabilities are regarded as the ground-truth values for the training process.

As for the graph transformer training stage, $\mathcal{G}$ is fed into the graph transformer, and predictions $P'_{leakage}$ and $P'_{blockage}$ are derived as the output of the graph transformer. Losses between the predictions and the corresponding ground-truth values are used to train the graph transformer. When the training is converged, the graph transformer can efficiently give accurate predictions about defects probabilities of different designs.

V. Experimental Results

A. Reliability Evaluation of Existing Fault-Tolerant Designs

With the help of the proposed quantification models, we are able to objectively and quantitatively evaluate the reliabilities of some classic designs and their fault-tolerant modifications. As shown in Fig. 9, designs in the first row are the fault-tolerant transportation channel, original CoMUX, and 8-to-4

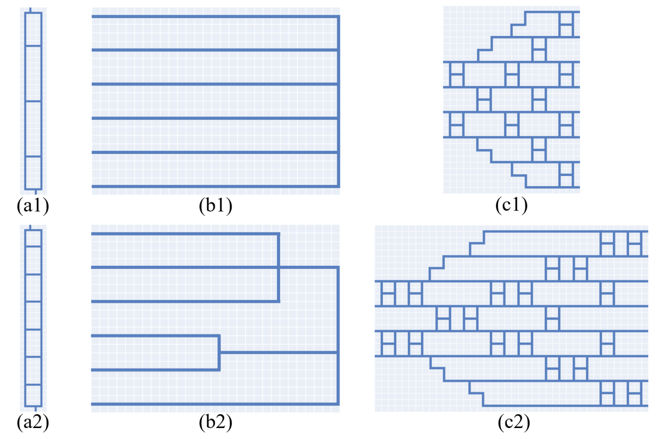


Fig. 9: Classic designs with their modifications. (a1) Original fault-tolerant transportation channel and (a2) enhanced fault-tolerant transportation channel [11]. (b1) Original CoMUX and (b2) optimized CoMUX [13]. (c1) 8-to-4 crossbar and (c2) expanded 8-to-4 crossbar [12].

crossbar, respectively. Designs in the second row are modifications of these classic designs [11]–[13].

We do not artificially define backup channels based on the functions of these designs but analyze the topology of these designs. The proposed quantification models are applied with three different pb_{unit} and three different pl_{unit} to evaluate the reliabilities of these classic designs and their modifications at different precision manufacturing processes. According to the results shown in TABLE II, all three modifications outperform the original designs in the reliability against blockage, and design (b2) also improves the reliability against leakage, which are consistent with the description in the corresponding papers [11]–[13]. There is no mention of leakage resistance in the papers for the other two modifications, and they are tested to be more prone to leakage than the original designs due to the increased total length and density of channels, which is consistent with the law summarized in [6]. We conclude that it is the use of channel merging in design (b2) that allows the design to improve in both reliabilities against blockage and leakage, and the use of back-up channels in design (a2) and (c2) can improve reliability against blockage but increases the risk of leakage.

B. Performance of the Graph-Transformer-based Implementation

As mentioned in Section IV C, we have generated 26411 designs as the dataset. These designs are divided into three

TABLE II: Reliabilities of Designs in Fig. 9

Design	$P_{blockage}$			$P_{leakage}$		
	pb_{unit} =0.1%	pb_{unit} =0.3%	pb_{unit} =0.5%	pl_{unit} =10%	pl_{unit} =20%	pl_{unit} =30%
(a1)	0.11%	0.35%	0.64%	0.33%	11.45%	65.82%
(a2)	0.10%	0.32%	0.57%	0.37%	12.98%	70.66%
(b1)	19.44%	47.74%	66.13%	1.12%	34.43%	97.74%
(b2)	11.83%	31.78%	47.53%	0.85%	28.09%	95.15%
(c1)	1.21%	3.68%	6.22%	1.92%	54.15%	99.93%
(c2)	0.40%	1.22%	2.06%	3.57%	76.49%	99.99%

[1] (a1) is an original design in [11] and (a2) is the modified design in [11]; (b1) is the original design in [13], and (b2) is the modified design in [13]; (c1) is an original design in [12] and (c2) is the modified design in [12].

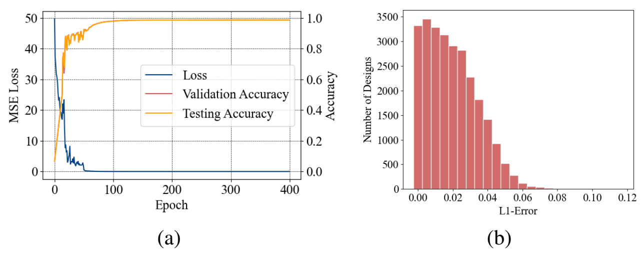


Fig. 10: Training effects of the graph transformer. (a) Loss and accuracy curves. (b) L1-error distributions.

parts: 15847 for training, 5282 for validation, and 5282 for testing. We use the proposed quantification models to calculate these designs' leakage and blockage probabilities, which are used as supervised labels. We train the graph transformer on a server with one Intel i7-9700K CPU and two Nvidia GeForce RTX 2080 Ti GPUs for 500 epochs and choose the converged model with the highest accuracy. To improve the robustness of the training process and get better training effects, we add noise to the training data and labels during the first 100 epochs [24]. Adam [25] is chosen as the optimizer, and MSE is selected as the loss function. The learning rate is set to 0.001 initially and is halved every 50 epochs. The batch size is set to 256. The dropout probability for the pre-encoder and the predicting block is set to 0.5.

Fig. 10 (a) shows the training curves of the graph transformer. The training converges quickly, and the average converged accuracy of $P'_{blockage}$ and $P'_{leakage}$ on the set of testing designs is 98.7%. As shown in Fig. 10 (b), the average L1-error of the graph transformer is 0.022, and the L1-error distribution is centered on the small values, proving the accurate learning performance of the graph transformer.

To show the efficiency of the graph transformer, we change the granularity of the same design from $1\mu m$ to $25\mu m$, and test the time usage and accuracy of the proposed quantification models and the graph-transformer-based implementation. As shown in Fig. 11 (a), as the granularity gets rougher and rougher, due to the increased distortion between the meshed representation and the actual design, the accuracy given by directly applying the proposed quantification models decreases from 0.998 to 0.955, and the corresponding time usage also decreases significantly from $9987.382s$ to $0.063s$. In stark contrast, as shown in Fig. 11 (b), thanks to the operation of

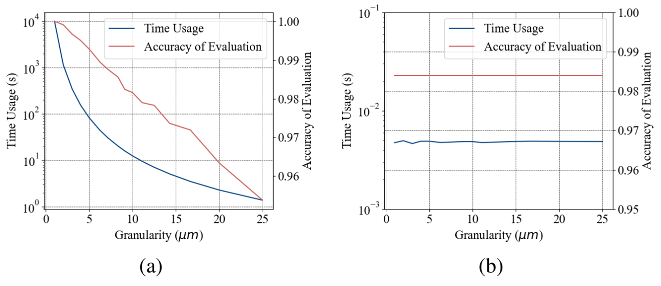


Fig. 11: Time usage and accuracy of evaluations with different granularities. (a) Directly apply the proposed quantification models. (b) Apply the graph-transformer-based implementation.

TABLE III: Performances of Fitting for pb_{unit}

pb_{unit}	Error of MAE	Error of Weighted MAE	Improvement
0.1%	0.08%	0.07%	12.50%
0.2%	0.78%	0.76%	2.57%
0.3%	1.94%	1.79%	7.73%
0.4%	2.47%	2.39%	3.24%
0.5%	2.96%	2.87%	3.05%

TABLE IV: Performances of Fitting for pl_{unit}

pl_{unit}	Error of MAE	Error of Weighted MAE	Improvement
10%	18.31%	18.27%	0.22%
15%	12.19%	12.12%	0.57%
20%	8.55%	8.40%	1.75%
25%	6.88%	6.81%	1.02%
30%	5.66%	5.45%	3.71%

abstracting the design into a graph in terms of key grid points, though dealing with designs with different granularities, the graph-transformer-based implementation has consistent accuracy at around 0.983, while the time usage is always around $0.0046s$.

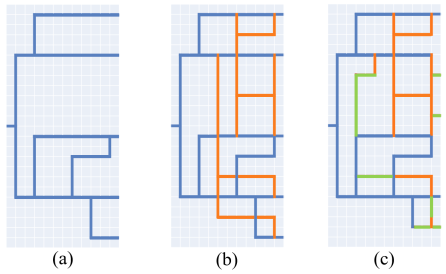
C. Performance in Adaptability

To verify the adaptability of the proposed quantification models, we change pb_{unit} from 0.1% to 0.5%, and pl_{unit} from 10% to 30%, and apply random experiments to test the performance of the adaptation approach proposed in Section III C. In order to reduce the bias caused by randomness in every fitting, we repeat 2000 fittings and calculate the average errors by using MAE and the proposed weighted MAE for each pb_{unit} and pl_{unit}, respectively.

In each fitting, following the adaptation approach proposed in Section III C, we set $n = 5$, $l_s = \{200, 400, 600, 800, 1000\}\mu m$, $d_s = \{2, 4, 6, 8, 10\}\mu m$. We generate $N_{total} = 500$ testing designs for each l_s and d_s, which have $pb_{unit}(l_s)$ and $pl_{unit}(d_s)$ chance to fail, respectively. We record the number of failed testing designs in blockage and leakage testing sets as $N_b(l_s)$ and $N_l(d_s)$, respectively. The statistical data is then calculated by Eq. 7. In the end, we perform curve fitting based on the statistical data using MAE and the proposed weighted MAE, whose weights are set as $\{0.08, 0.12, 0.20, 0.28, 0.32\}$. The fitting performances are shown in TABLE III and TABLE IV. According to the results, both MAE and the proposed weighted MAE can achieve accurate curve fitting with at most 2.96% error for pb_{unit} and 18.31% error for pl_{unit}. Moreover, the proposed weighted MAE can improve the fitting accuracy by at least 2.57% and 0.22% for pb_{unit} and pl_{unit} respectively. Hence, the proposed quantification models are verified to be adaptive for different manufacturing processes.

D. Example of a Design Aided by the Proposed Quantification Models

To show how the proposed quantification models can provide timely feedback to help improve microfluidic designs, we take the fault-tolerant design proposed in [10] as an example. As shown in Fig. 12(b), blue lines represent the original design, and orange lines represent channels proposed to be added to realize fault tolerance in [10]. Based on our reliability quantification models, we find that compared to the original design, the fault-tolerant design proposed in [10] improves the reliability against blockage by 10.6%, but slightly decreases the reliability against leakage by 1.6%.



Design	$P_{blockage}$	$P_{leakage}$
Original Design	28.0%	5.3%
Design with Fault Tolerance	17.4%	6.9%
Design with Further Modification	15.7%	1.9%

Fig. 12: An example of improving existing fault-tolerant design using the proposed quantification model. (a) Original design. (b) Fault-tolerant design proposed in [10]. (c) Design with further modification.

We then randomly modify the fault-tolerant design by adding, removing, and changing the positions of channel segments within a small range that guarantees the original functionality of the design. Once a randomly modified design is generated, we analyze its reliability with the graph-transformer-based implementation of our proposed quantification models to see if the modification can bring improvement. Fig. 12 (c) shows one randomly modified design, in which green lines indicate our changes. Compared to the original fault-tolerant design, the new design further reduces $P_{blockage}$ and $P_{leakage}$ from 17.4% to 15.7% and 6.9% to 1.9%, hence improvements of 9.8% and 72.5%, respectively.

It is noteworthy that Fig. 12 (c) is just one of the many possible modified designs, and we have yet to apply any optimization to derive the best-modified design but are already able to improve the reliability significantly. Hence, the great potential for design reliability from the efficient feedback that our quantification models can provide is evident.

VI. CONCLUSION

In this paper, we proposed the first quantification model that can easily be adapted to different manufacturing processes by changing a few parameters, and can be used to evaluate the reliabilities of different microfluidic designs. Moreover, we proposed a graph-transformer-based implementation of the quantification models, which can conduct rapid evaluations while maintaining a relatively high accuracy of evaluation. According to the experimental results, with the help of the efficient feedback provided by the proposed quantification models, even a random modification can already improve reliabilities against blockage and leakage of the existing fault-tolerant design by 9.8% and 72.5%, respectively. Hence, the proposed quantification models and their graph-transformer-based implementation promise to support us in developing automatic reliability modification tools to improve the reliability of LoC designs in the future.

References

[1] S. J. Maerkl, T. A. Thorsen, X. Bao, S. R. Quake, and V. Studer, "Microfluidic large scale integration," Dec. 5 2006. US Patent 7,143,785.

[2] T. H. Schulte, R. L. Bardell, and B. H. Weigl, "Microfluidic technologies in clinical diagnostics," *Clinica Chimica Acta*, vol. 321, no. 1-2, pp. 1–10, 2002.

[3] G. M. Whitesides, "The origins and the future of microfluidics," *nature*, vol. 442, no. 7101, pp. 368–373, 2006.

[4] D. Mark, S. Haeberle, G. Roth, F. v. Stetten, and R. Zengerle, "Microfluidic lab-on-a-chip platforms: requirements, characteristics and applications," *Microfluidics based microsystems*, pp. 305–376, 2010.

[5] K. Hu, T.-Y. Ho, and K. Chakrabarty, "Test generation and design-for-testability for flow-based mvlsi microfluidic biochips," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, pp. 1–6, IEEE, 2014.

[6] K. Hu, F. Yu, T.-Y. Ho, and K. Chakrabarty, "Testing of flow-based microfluidic biochips: Fault modeling, test generation, and experimental demonstration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1463–1475, 2014.

[7] M. Li, Y. Zhang, J. Y. Lee, H. Gasvoda, I. E. Araci, T.-M. Tseng, and U. Schlichtmann, "Integrated test module design for microfluidic large-scale integration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2022.

[8] F. Su, K. Chakrabarty, and V. K. Pamula, "Yield enhancement of digital microfluidics-based biochips using space redundancy and local reconfiguration," in *Design, Automation and Test in Europe*, pp. 1196–1201, IEEE, 2005.

[9] T. C. Cameron, A. Randhawa, S. M. Grist, T. Bennet, J. Hua, L. G. Alde, T. M. Caffrey, C. L. Wellington, and K. C. Cheung, "Pdms organ-on-chip design and fabrication: Strategies for improving fluidic integration and chip robustness of rapidly prototyped microfluidic in vitro models," *Micromachines*, vol. 13, no. 10, p. 1573, 2022.

[10] Y. Zhu, B. Li, T.-Y. Ho, Q. Wang, H. Yao, R. Wille, and U. Schlichtmann, "Multi-channel and fault-tolerant control multiplexing for flow-based microfluidic biochips," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, IEEE, 2018.

[11] W.-L. Huang, A. Gupta, S. Roy, T.-Y. Ho, and P. Pop, "Fast architecture-level synthesis of fault-tolerant flow-based microfluidic biochips," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 1667–1672, IEEE, 2017.

[12] Y. Moradi, M. Ibrahim, K. Chakrabarty, and U. Schlichtmann, "Fault-tolerant valve-based microfluidic routing fabric for droplet barcoding in single-cell analysis," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1484–1487, IEEE, 2018.

[13] S. Liang, M. Li, T.-M. Tseng, U. Schlichtmann, and T.-Y. Ho, "Comux: Combinatorial-coding-based high-performance microfluidic control multiplexer design," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.

[14] B. Jiang, H. Guo, D. Chen, and M. Zhou, "Microscale investigation on the wettability and bonding mechanism of oxygen plasma-treated pdms microfluidic chip," *Applied Surface Science*, vol. 574, p. 151704, 2022.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[16] D. Q. Nguyen, T. D. Nguyen, and D. Phung, "Universal graph transformer self-attention networks," in *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion)*, 2022.

[17] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, pp. 2704–2710, 2020.

[18] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-supervised graph transformer on large-scale molecular data," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12559–12571, 2020.

[19] W. Ji, X. Guo, S. Pan, T.-Y. Ho, U. Schlichtmann, and H. Yao, "Gnn-based concentration prediction for random microfluidic mixers," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, (New York, NY, USA), p. 763–768, Association for Computing Machinery, 2022.

[20] W. Ji, T.-Y. Ho, J. Wang, and H. Yao, "Microfluidic design for concentration gradient generation using artificial neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2544–2557, 2019.

[21] J.-J. Greffet, P. Bouchon, G. Brucoli, E. Sakat, and F. Marquier, "Generalized kirchhoff law," *arXiv preprint arXiv:1601.00312*, 2016.

[22] V.-A. Nguyen, D. Q. Nguyen, V. Nguyen, T. Le, Q. H. Tran, and D. Phung, "Regvd: Revisiting graph neural networks for vulnerability detection," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, pp. 178–182, 2022.

[23] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, pp. 2048–2057, PMLR, 2015.

[24] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning*, pp. 1310–1320, PMLR, 2019.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.