A Unified Framework for Layout Pattern Analysis with Deep Causal Estimation

Ran Chen[1], Shoubo Hu[2], Zhitang Chen[2], Shengyu Zhu[2], Bei Yu[1], Pengyun Li[3], Cheng Chen[3], Yu Huang[3], Jianye Hao[2]

[1]The Chinese University of Hong Kong
[2]Huawei Noah's Ark Lab
[3]HiSilicon

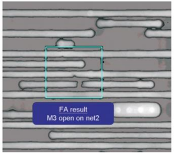
Nov. 1, 2021

HUAWEI

Outline

1. Background
2. Previous works & Our framework
3. Algorithm
4. Result

- By analyzing multiple layout-aware diagnosis reports to identify the underlying systematic defect distribution.
- Identify the root cause[1] in short time is important. LPA reduces the cycle time of physical failure analysis (PFA) from months to days.



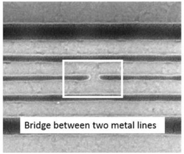
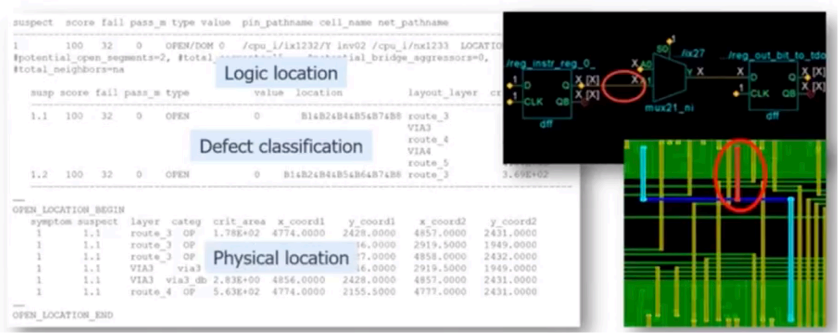


Image of Open/Bridge defects.

[1] Root cause: most critical systematic defect issue that has maximum impact on yield.

Layout-aware diagnosis results

- Physical call-outs provide valuable information for yield analysis[2].
- Reduced suspect area accelerates failure analysis.



[2] Source:https://resources.sw.siemens.com/en-US/fact-sheet-tessent-yieldinsight-factsheet

- It is not clear how diagnosis report generated. (black box)
- Diagnosis results have ambiguity.
  - Multiple suspect patterns ($10^2 \sim 10^4$ clips in one report).
  - Difficult to use raw diagnosis results to produce an accurate defect distribution or select best die for failure analysis.

◐ E.g. Multiple suspects in one netlist.

```
suspect  score fail_match pass_mismatch type          value  pin_pathname cell_name net_pathname layout_status
---------------------------------------------------------------------------------------------
1        100   9          0             OPEN/DOM      both   .../n_6950
         #potential_open_segments=1, #total_segments=1,  #potential_bridge_aggressors=3, #total_neighbors=22
         suspect  score fail_match pass_mismatch type          value  location layout_layer critical_area
         -------------------------------------------------------------------------------------
         1.1      100   9          0             DOM           aggr   .../n_4643  1 Metal3        4.85E+06
         1.2      96    9          2             OPEN          both   B1          2 Metal1        4.82E+04
                                                                                  3 via1          5.79E+04
                                                                                  4 Metal2        2.56E+05
                                                                                  5 via2          9.87E+04
                                                                                  6 Metal3        4.44E+05

2        100   9          0             OPEN/DOM      both   .../p0207A67923/Y MXI2X1
         #potential_open_segments=1, #total_segments=1,  #potential_bridge_aggressors=0, #total_neighbors=15
         suspect  score fail_match pass_mismatch type          value  location layout_layer critical_area
         -------------------------------------------------------------------------------------
         2.1      100   9          0             CELL          both   .../p0207A67923 7
```

☹ A 'supervised' learning task with a mass of noise.

Problem Definition

- The objective of LPA in this work is to identify true root cause(s) of systematic defect by analyzing a dataset consisting of m diagnosis reports $R = \{r^e\}_{e=1}^{m}$ and layout snippets of potential root causes in these reports.

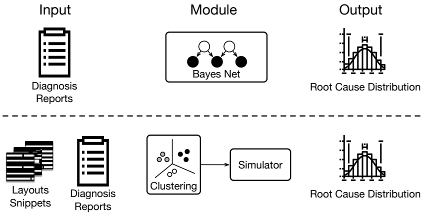
Problem Definition

- Each report r^e consists of several independent symptoms (i.e., defects), whose possible causes are also given along with several important properties (e.g., ID, score, etc.).

Table: Notation on Diagnosis Report Features.

Feature	**Description**
rule_id	ID of the rule of the violation
s_i	The score of suspect i reported in the diagnosis report
h_i	DFM hits of suspect i
v_i	DFM violations of suspect i
$\langle x_i, y_i \rangle$	Location of suspect i in designs
M1	Layer name of suspect
OPEN	Defect category

Previous works



- Upper[ITC’12][3] [ETS’17][4]: No consideration on root cause layout patterns which largely restricts their applicability to real tasks.
- Lower[TCAD’15][5], [ITC’10][6]: Resolution is limited, a failure analysis expert’s judgment is required to pick a single layout snippet for each cluster.

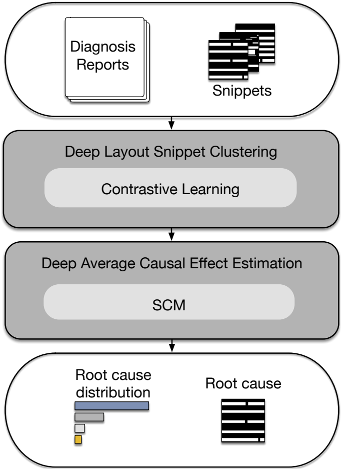
[3] Brady Benware et al. (2012). “Determining a failure root cause distribution from a population of layout-aware scan diagnosis results”. In: *IEEE Design & Test of Computers* 29.1, pp. 8–18.

[4] Wu-Tung Cheng, Yue Tian, and Sudhakar M Reddy (2017). “Volume diagnosis data mining”. In: *2017 22nd IEEE European Test Symposium (ETS)*. IEEE, pp. 1–10.

[5] Wing Chiu Jason Tam and Ronald D Shawn Blanton (2015). “LASIC: Layout analysis for systematic IC-defect identification using clustering”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.8, pp. 1278–1290.

[6] Wing Chiu Tam, Osei Poku, and Ronald D Blanton (2010). “Systematic defect identification through layout snippet clustering”. In: *2010 IEEE International Test Conference*. IEEE, pp. 1–10.

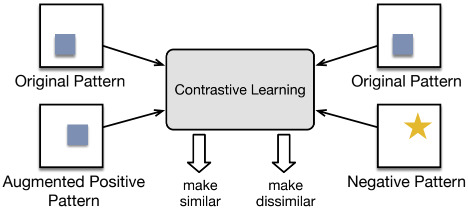
Our Framework



An overview of the framework.

- We propose a unified solution to volume diagnosis-based root causes layout pattern identification task. Both pattern clustering and root cause identification are taken into consideration. Our framework can identify the critical root causes and provide high-resolution clustered snippets for further analysis.

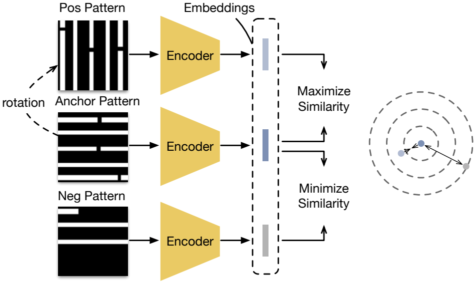
Contrastive learning



- Maximize the similarity between latent features of a pattern and its augmented version and simultaneously minimize the similarity between latent features of inputs correspond to different original patterns.

Deep Layout Snippet Clustering

- Contrastive learning based clustering
  - Equivalent snippets share the unique latent code.
  - Converting n snippets into latent codes and perform conventional k-mean algorithm on the latent codes. Return distance matrix $\mathbf{D} \in \mathbb{R}^{n \times k}$.

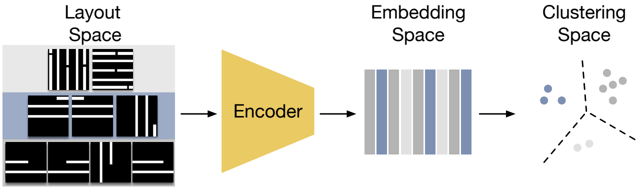


☺ Improvement on resolution: equivalent snippets (shift, rotation and mirror) are clustered in same group.

- Distance matrix **D** to membership matrix **P**.

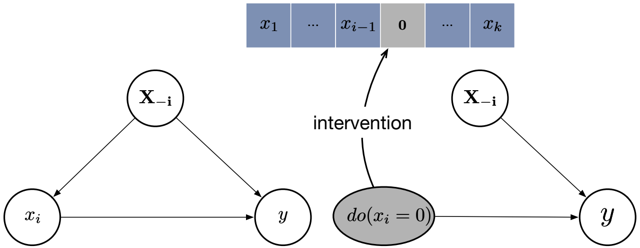
$$[\mathbf{P}]_{j,i} = \frac{\exp\left(-\mathbf{D}_{j,i}/\tau\right)}{\sum_{i'} \exp\left(-\mathbf{D}_{j,i'}/\tau\right)}, \tag{1}$$

  - The layout snippets closer to the cluster center have higher probabilities.
  - Compression: from an image to a point.



A Demo on Deep Layout Snippet Clustering.

- Build the Structural Causal Model (SCM) between candidate layout patterns and root cause(s).
- Use Average Causal Effect (ACE) estimation to identify true root cause(s) from a large amount of potential root causes using diagnosis reports and the results of layout pattern matching.



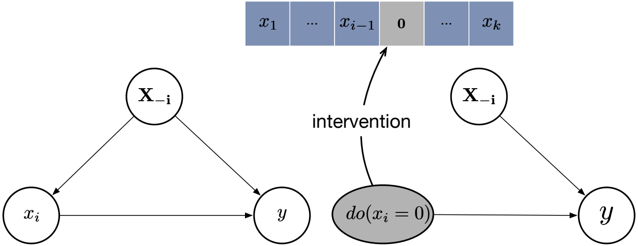
Left: The defect SCM for Layout Pattern Analysis without intervention. Right: Apply intervention on cluster i.

[6] J. Pearl, Causality. Cambridge university press, 2009

This ACE can be estimated as:

$$ACE^{y}_{do(x_i)} = \left|\mathbb{E}[y|do(x_i = 0)] - \mathbb{E}[y|do(x_i = 1)]\right| . \tag{2}$$

- The ACE of x_i on y characterizes the causal effect of the presence of layout pattern x_i on the systematic defect.



Left: The defect SCM for Layout Pattern Analysis without intervention. Right: Apply intervention on cluster i.

[6] We assume that the true root cause has the most significant ACE on the systematic defect.

Deep Average Causal Effect Estimation

- An Multilayer Perceptron (MLP) to characterize the causal relationship between candidate layout patterns and systematic defect.
- Neural network attribution [ICML2019][7] is used to speed up the inference:

$$\mathbb{E}[y|do(x_i=0)] \approx f'(\boldsymbol{\mu}_{i0}) + \\ \frac{1}{2}\mathrm{tr}(\nabla^2 f'(\boldsymbol{\mu}_{i0})\mathbb{E}[(l_{in}-\boldsymbol{\mu}_{i0})(l_{in}-\boldsymbol{\mu}_{i0})^T|do(x_i=0)]), \tag{3}$$

[7] Aditya Chattopadhyay et al. (2019). "Neural network attributions: A causal perspective". In: *International Conference on Machine Learning*. PMLR, pp. 981–990.

Experimental settings

- We adopt *defect injection* [ITC'12][1] experiments to evaluate the performance of our framework.
- Three scenarios are conducted
  1. Single root cause.
  2. Single root cause with random injection noise.
  3. Multiple root causes with noise.
- Inference on single NVIDIA V100 GPU.
- A diagnosis statistical approach is presented as the baseline.

Table: Layout Design Information.

	Size ($\mu m \times \mu m$)	#Layers	#Gates
Case 1	8881×9328	5	9337
Case 2	429×384	9	1560k
Case 3	8033×7822	6	9278k

[1] Brady Benware et al. (2012). "Determining a failure root cause distribution from a population of layout-aware scan diagnosis results". In: *IEEE Design & Test of Computers* 29.1, pp. 8–18.

Results

- Scenario 1: single root cause.

Table: Accuracy(%) on Noise-free Datasets.

Dataset	`Baseline`	`Commercial Tool`	`Ours`
Case 1	25.00	98.53	**100.00**
Case 2	55.88	92.52	**98.04**
Case 3	58.06	98.92	98.92
Average	46.31	96.66	**98.99**

- Scenario 2: single root cause with random injection noise.

Table: Accuracy(%) on Noisy Case 2 Datasets.

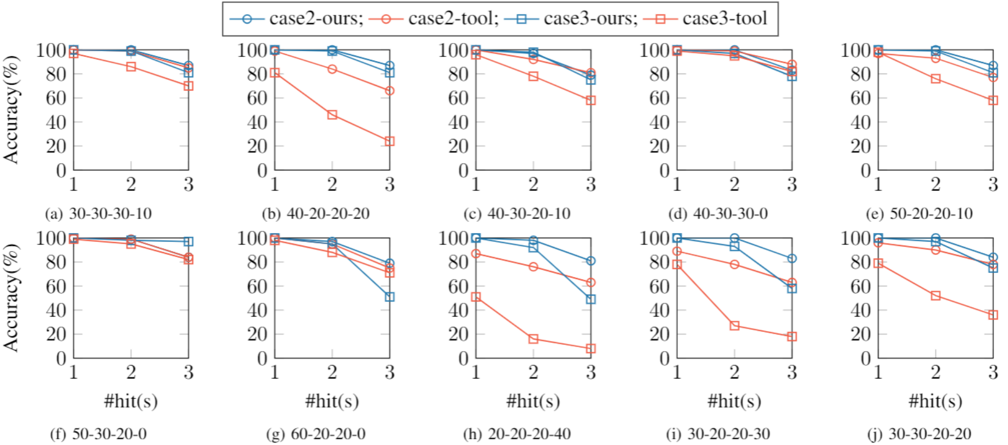
Noise(%)	Baseline	Commercial Tool	Ours
80	19.57	84.11	**97.83**
70	37.62	92.52	**95.05**
60	44.55	94.39	**98.02**
50	49.02	94.39	**96.08**
40	50.98	93.45	**95.10**
30	51.96	**93.45**	93.14
20	58.82	92.52	**95.10**
10	55.88	93.46	**98.04**
Average	46.05	92.29	**96.05**

- Scenario 3: multiple root causes with noise.

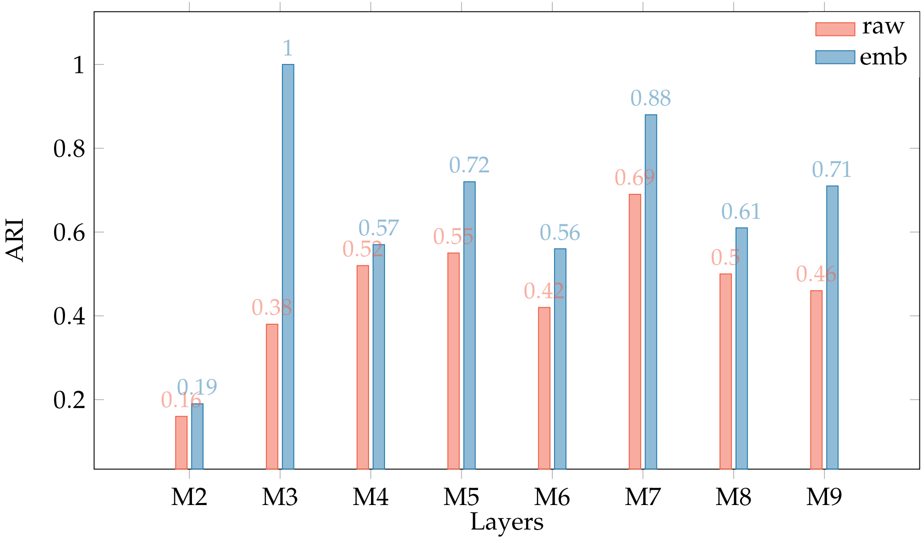
Table: Accuracy(%) on Mixture Datasets.

Proportion (r1%-r2%-r3%-noise%)	Commercial Tool		Ours	
	Case 2	Case 3	Case 2	Case 3
30-30-30-10	85	70	**87**	**81**
40-20-20-20	66	24	**73**	**74**
40-30-20-10	**81**	70	79	**75**
40-30-30-00	**88**	**82**	83	78
50-20-20-10	**77**	**58**	76	58
50-30-20-00	84	**82**	84	79
60-20-20-00	75	**71**	**79**	50
20-20-20-40	63	8	**81**	**49**
30-20-20-30	63	18	**83**	**58**
30-30-20-20	78	36	**84**	**75**
Average	76	52	**81**	**68**

Results



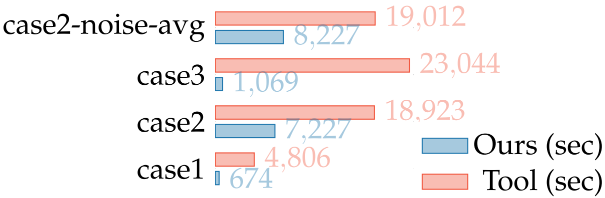
Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture datasets.



ARI of conducting layout pattern matching using raw layout snippets and embeddings.

Inference time

- We get $\times 8.4$ speedup on average at inference.



THANK YOU!