Machine Intelligence on Layout Defect Identification

Ran CHEN

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
June 2022

Thesis Assessment Committee

Professor YOUNG Fung Yu (Chair)
Professor YU Bei (Thesis Supervisor)
Professor XU Qiang (Committee Member)
Professor YANG Fan (External Examiner)

Abstract of thesis entitled:

Machine Intelligence on Layout Defect Identification

Submitted by Ran CHEN

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in June 2022

Recent years have seen an explosion of interest in Design for Manufacturability (DFM) and Artificial Intelligence (AI). The interest is directly attributed to the difficulties of manufacturing integrated circuits in nanometer-scale CMOS technologies with high functional and parametric yield. Especially the systematic defects in layout designs. This thesis focuses on AI solutions for layout defect identification (LDI), including layout hotspot detection, layout pattern analysis, and root cause identification.

With the development of the semiconductor industry, transistor feature size shrinks rapidly, which significantly challenges manufacturing yield. For instance, the low-fidelity pattern on the wafer (a.k.a. "hotspot") is one of the emerging issues in manufacturing. To ensure the printability of layout designs, an efficient and accurate hotspot detector is indispensable. Developed deep learning techniques have recently shown their superiorities on hotspot detection tasks. Existing hotspot detectors can only

handle defect detection from one small layout clip each time, this may be very time-consuming when dealing with a large full-chip layout. We develop a new end-to-end framework that can detect multiple hotspots in a large region at a time and promise a better hotspot detection performance. We design a joint auto-encoder and inception module for efficient feature extraction. A two-stage classification and regression framework are designed to detect hotspots with progressive accurate localization, which provides a promising performance improvement.

A high and stable yield could ensure the profitability and reliability of products. Specific layout patterns that are hard to fabricate tend to cause more systematic defects, such as open or bridge defects in neighboring wires. These layout patterns are an important source of yield loss. Since layout configurations of new designs may differ from existing ones, identifying layout patterns that lead to yield loss through test chips, SRAMs, etc., is becoming less effective. Physical failure analysis (PFA) is a straightforward method to determine whether a layout pattern is the root cause of systematic defects. Fabrication is often time-consuming and expensive, and requires both experience and a thorough understanding of the fabrication process. A unified framework for layout pattern analysis with deep causal effect estimation is proposed. A causal relationship between the root causes and the structural defect is described in our framework.

To further improve the robustness of root cause identification and eliminate the noise in diagnosis reports. We develop a

reinforcement learning agent to detect the root cause efficiently with high accuracy. A reinforcement learning method based on actor-critic is proposed to identify the optimal structural causal graph according to a reward function based on Average Causal Effects. At the inference stage, the reinforcement learning agent is forward to predict the causal graph, then the Average Causal Effect will be calculated to infer the root cause distribution.

Based on experimental results, we demonstrate that the deep learning methods we propose are efficient and effective at identifying defects in layouts. As the manufacturing industry demands more quality and quantity, we hope these algorithms and frameworks can spur the industry forward.

摘要

近年來，人們對可製造性設計 (DFM) 和人工智能 (AI) 的興趣激增。這直接歸因於在具有高性能和受參數影響良率的納米級 CMOS 技術中製造集成電路的困難。尤其是版圖設計中的系統性缺陷。本論文專注於佈局缺陷識別（LDI）的人工智能解決方案，包括佈局熱點檢測、佈局模式分析和根因識別。

隨著半導體產業的發展，晶體管特徵尺寸迅速縮小，這極大地挑戰了製造良率。例如，晶圓上的低保真圖案（又名 “熱點”）是製造中新出現的問題之一。為確保版圖設計的可印刷性，高效準確的熱點檢測器必不可少。開發的深度學習技術最近在熱點檢測任務中顯示出它們的優勢。現有的熱點檢測器每次只能從一個小的佈局剪輯中進行缺陷檢測，這在處理大型全芯片佈局時可能非常耗時。我們開發了一個新的端到端框架，可以一次檢測大區域中的多個熱點，並承諾更好的熱點檢測性能。我們設計了一個聯合自動編碼器和初始模塊，用於高效的特徵提取。一個兩階段分類和回歸框架用來檢測具有漸進準確定位的熱點，這提供了有保障的性能改進。

高而穩定的產量可以保證產品的盈利能力和可靠性。難以製造的特定佈局圖案往往會導致更多系統缺陷，例如相鄰導線中的開路或橋接缺陷。這些佈局圖案是良率損失的重要來源。

由於新設計的佈局配置可能與現有設計不同，通過測試芯片、SRAM 等識別導致良率損失的佈局模式變得越來越低效。物理失效分析 (PFA) 是一種確定佈局模式是否是系統缺陷根本原因的直接方法。製造通常既耗時又昂貴，並且需要經驗和對製造過程的透徹了解。我們提出了一個具有深度因果效應估計的佈局模式分析的統一框架，該框架描述了根本原因和結構缺陷之間的因果關係。

進一步提高根本原因識別的穩健性，消除診斷報告中的噪音。我們開發了一種強化學習代理模型，以高精度高效地檢測根本原因。提出了一種基於 actor-critic 的強化學習方法，根據基於平均因果效應的獎勵函數識別最優結構因果圖。在推理階段，強化學習代理向前預測因果圖，然後計算平均因果效應來推斷根本原因分佈。

基於實驗結果，我們證明了我們提出的深度學習方法在識別佈局缺陷方面是有效的。隨著製造業對質量和數量的要求越來越高，我們希望這些算法和框架能夠推動行業向前發展。

Acknowledgement

I would like to express my deepest gratitude to my advisor, Prof. Bei YU, whose sincerity and encouragement I will never forget. This thesis would not have been possible without Dr. Zhitang Chen, whose guidance from the initial step of research enabled me to develop an understanding of the subject. I am thankful for the extraordinary experiences he arranged for me and for providing opportunities for me to grow professionally.

It has been a great honor to work with many great collaborators during my Ph.D. studies. Very special thanks to Prof. Fan Yang for plenty of cooperation on DFM topics. Thanks to Dr. Shoubo Hu and Dr. Yu Huang for fascinating collaborations and suggestions on the topic of layout pattern analysis. Thanks to hao GENG, haoyu Yang, and Yuzhe Ma for constructive discussions and cooperation. I would like to thank other brilliant collaborators, Dr. Shu Liu, Dr. Xuan Zeng, Dr. Shengyu Zhu, Prof. Jianye Hao, Cheng Chen, Pengyun Li, Binwu ZHU, and all other CUDA members.

I am grateful for my parents whose constant love and support keep me motivated and confident. My accomplishments and

success are because they believed in me. I am forever thankful for the unconditional love and support throughout the entire thesis process and every day.

This work is dedicated to my ever supportive and loving family. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Challenges and Motivations

To illustrate how Artificial Intelligence methodologies help academia and industry address the specific defect issues in manufacturing, we present the current research status of several layout defect identification (LDI) applications. Layout defects can be identified and fixed by uncovering and eliminating the common root causes.

Hotspot Detection. With the shrinking VLSI technology node, a large gap exists between the mask pattern feature size and lithography system wavelength. Light diffraction seriously decreases the printability of the mask layout, and circuit failures (open circuits or short circuits) are more likely to occur for some patterns. Therefore it is necessary to detect and correct the problematic patterns (i.e. hotspots) before the manufacturing process. A variety of resolution enhancement technologies (RETs) have been developed to provide yielding-friendly

patterns under sub-wavelength lithography conditions. The effectiveness of RETs, however, depends on a number of factors, including pattern compatibility with OPC algorithms and pitch differences. Adopting them alone does not guarantee that you will print the ideal shapes, so additional layout refinements are needed to increase the quality of printed patterns. Methodologies such as pattern matching-based [92, 97, 113], machine learning model-based [58, 66, 84, 106] hotspot detection algorithms are proposed. In comparison with lithographic simulation, pattern matching is faster. It is effective for detecting already known or similar patterns, while recognition of unknown patterns is poor.

Recent deep learning techniques [6, 12, 15, 33, 43, 102] have demonstrated superiority in hotspot detection tasks. In order to ensure printability of layout designs, a hotspot detector with high accuracy and low false alarms is essential. cin identifying hotspots benefitting from the development of Artificial Intelligence. However, these hotspot detectors [102, 108] only work on small clips extracted from a whole chip layout and can only detect one hotspot location at a time that occurs at a center (i.e. core in [91]) of each clip. Conventional hotspot detection schemes require repeatedly scanning overlapping regions of a full chip design. It could be a waste of computational resources and time-consuming when facing with extremely large layouts. It is thus crucial to find an efficient way to detect hotspots on large designs while saving resources.

Root Causes Identification. Performing hotspot detec-

tion on entire layouts may result in *overcorrection* which can adversely affect chip area and performance. Most hotspots are candidates, that may not happen as the actual physical deformation leads to malfunction. Identifying the root cause of systematic defects takes one step further than detecting hotspots, which is a more challenging objective. The root cause refers to a difficult-to-manufacture layout structure that becomes vulnerable to failure. Feature sizes on layouts shrink dramatically to smaller than lithography wavelengths, which makes them increasingly susceptible to failure. Determining what such features are is still an open question. During the early stages of manufacturing a new product, No matter whether a new manufacturing process is introduced for an existing design or a new design is added to an existing manufacturing process, there is usually a lower yield for the first batch of manufactured devices. In such cases, it is important to figure out why the yield is low. The yield engineers must identify defects, understand their causes, and modify the design or manufacturing process to improve yield. It is important to achieve an acceptable yield before volume production begins and to maintain it during volume production. Statistical techniques [3, 18] are being used to identify common defects from volume diagnosis reports. Utilizing an Expectation-Maximization algorithm [3], the optimal root cause distribution is learned by maximizing the likelihood of observed diagnosis reports. These methods do not consider root cause layout patterns, which limit their usefulness in real-

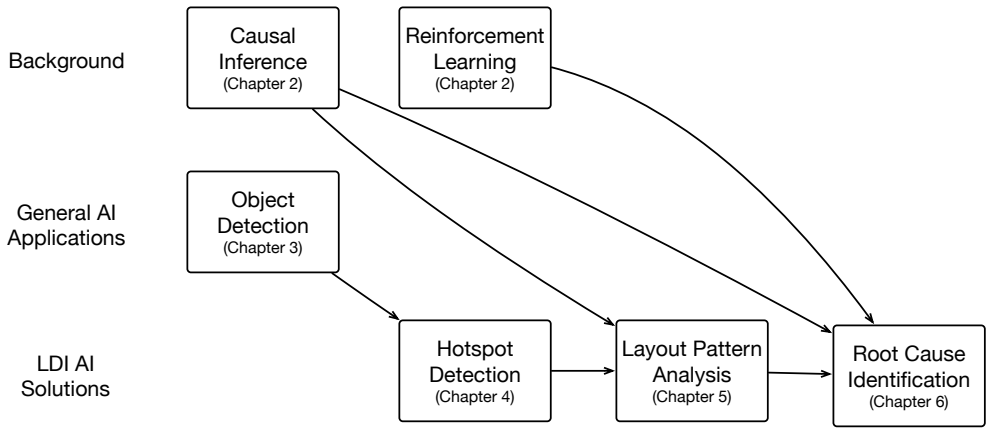


Figure 1.1 Organization chart of the thesis.

world situations. Also, the quality and diversity of diagnosis reports may be affected by the defect simulation process, which is not considered in these works. There is also a class of seminal works [85, 86] that emphasize the geometric structure of layout patterns by using clustering algorithms to improve IC-defect identification. While the resolution of the identification results of clustering-based methods is limited. Therefore, a methodology for identifying root causes with both high resolution and precision is essential.

1.2 Thesis Structure and Contributions

Throughout this thesis, we will examine a variety of methodologies for addressing the issues mentioned previously and explore the opportunities presented by AI technologies to improve yields. The overall organization of chapters as well as the dependencies is illustrated in Fig. 1.1.

The first contribution of the thesis is that a comprehensive survey and empirical studies on deep learning technology used in the article are provided. The background and related works of causal inference and reinforcement learning are introduced. This thesis proposes a general object recognition framework that recognizes daily objects and reveals the mechanism for their recognition from a macro perspective. The experimental results on several challenging datasets show that it has great potential to be used in defect identification tasks.

The second contribution of the thesis is that we develop a new end-to-end framework that can detect multiple hotspots in a large region at a time and promise a better hotspot detection performance. With the technology node of integrated circuits scaling down to 7nm and beyond, lithographic processes are supposed to manage the ever-shrinking size of the features. Unfortunately, the advancement of lithography techniques has lagged behind, lithographic process variations emerge during the manufacturing and thus lead to yield loss. Low fidelity patterns on a wafer represent one of the most pressing issues. In this thesis, a region-based hotspot detection framework is designed to speed up the detection efficiency. We design a joint auto-encoder and inception module for efficient feature extraction. A two-stage classification and regression framework is designed to detect hotspots with progressive accurate localization, which provides a promising performance improvement. Experimental results show that our framework enables a significant speed im-

provement over existing methods with higher accuracy and fewer false alarms.

The third contribution of the thesis is proposing a novel layout-aware diagnosis-based layout pattern analysis framework to identify the root cause efficiently. The yield of manufactured integrated circuits (ICs) is defined as the percentage of good dies among all dies manufactured. To ensure the profitability and reliability of products, a high and stable yield is crucial. However, as the feature size decreases, the underlying patterns in these layout designs are harder to fabricate and tend to generate more systematic defects, for example, open or bridge defects in neighboring wires. A unified layout pattern analysis framework is proposed to improve the resolution and accuracy simultaneously. At the first stage of the framework, an encoder network trained using contrastive learning is used to extract representations of layout snippets that are invariant to trivial transformations including shift, rotation, and mirroring, which are then clustered to form layout patterns. In the second stage, we model the causal relationship between any potential root cause layout patterns and the systematic defects by a structural causal model, which is then used to estimate the Average Causal Effect (ACE) of candidate layout patterns on the systematic defect to identify the true root cause. Experimental results on real industrial cases demonstrate that our framework outperforms a commercial tool with higher accuracies and around $\times 8.4$ speedup on average.

As the fourth contribution, a reinforcement-based root cause identification system is designed to further improve the performance. During the training, an actor-critic-based reinforcement learning method is proposed to learn to discover the optimal structural causal graph according to the reward correlated to the Average Causal Effect. At the inference stage, the converged reinforcement learning agent is forward to predict the causal graph, then the Average Causal Effect will be calculated correspondingly. Experimental results on several industrial cases show that the proposed reinforcement learning-based root cause identification system can provide accurate root cause distributions compared to the state-of-the-art frameworks.

The structure of the thesis is organized as follows. Chapter 2 provide related backgrounds and works about AI technologies used in LDI tasks. An overview of the recognition mechanism in artificial intelligence is provided in Chapter 3 using a general object detection framework. Chapter 4 covers the second contribution with corresponding technique details. Chapter 5 describes the third contribution about layout pattern analysis. Chapter 6 includes the fourth contribution which is an extension of LPA. Chapter 7 summarizes this thesis and delivers the possible future study directions.

□ **End of chapter.**

Chapter 2

AI Methodologies in LDI

In this chapter, we will introduce some AI methodologies that will be used in layout defect identification tasks. Primarily background knowledge about causal inference and reinforcement learning is included in this chapter. On causal inference, the reason why we need causality is introduced first. Following this, we provide a concise review of the structural causal model. The core components and mechanism of reinforcement learning are demonstrated.

2.1 Causal Inference

Causality is a generic relationship between an effect and the cause that gives rise to it. The definition of causality is hard to describe since the causes and effects we know about are intuitive in general. In contrast to statistical associations, establishing the causation is a crucial step toward human-level intelligence and can serve as a foundation for artificial intelligence.

Table 2.1 An example to show the Simpson's paradox.

Age \ Treatment	Treatment A	Treatment B
Young	234/270=87%	81/87=**92%**
Older	55/80=69%	192/263=**73%**
Overall	289/350=**83%**	273/350=78%

Causal inference is necessary - Simpson's paradox. According to TABLE 2.1, we can observe an interesting phenomenon called Simpson's Paradox, brought by the confounder. It can be observed that Medicine B has a higher recovery rate than Medicine A in both Young and Older patient groups. But when combining these two groups, Medicine A is the one with a higher recovery rate. When comparing the recovery rate in the whole group, most of the people taking medicine A are young, and the comparison shown in the table fails to eliminate the effects of age on the recovery rate. Frequency data is inappropriately interpreted causally. This can be resolved if confounding variables and causal relations are properly accounted for during the statistical analysis.

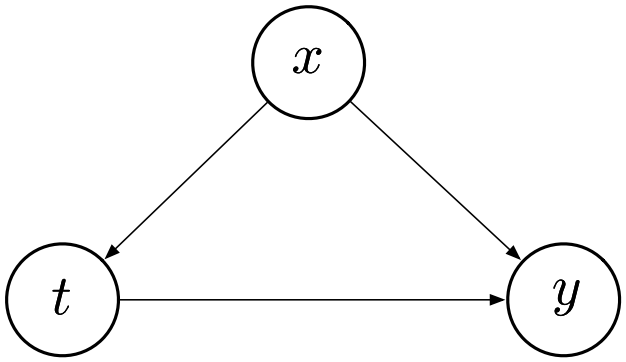
Structural Causal Model. Structural Causal Model (SCM) is a conceptual model that describes the causal mechanism of a system. An SCM normally consists of the causal graph and the structural equations. A causal graph forms a special class of Bayesian network with edges representing the causal effect. SCMs have a transformative impact on multiple data-intensive

disciplines (e.g. epidemiology, economics, etc.). They enable the codification of the existing knowledge in diagrammatic and algebraic forms and consequently leverage data to estimate the answers to interventional and counterfactual questions. To help readers understand how SCMs work, an example on Yelp rating is presented in Figs. 2.1 and 2.2, i.e., the restaurant category x(confounder), Yelp star rating t(treatment) and customer flow y(outcome).

- **Treatment** refers to the action that applied to the unit, which is the atomic research object in the treatment effect study.
- **Outcome** is the outcome of that treatment when applied on that unit is the potential outcome.
- **Confounders** are the variables that affect both the treatment and the outcome.

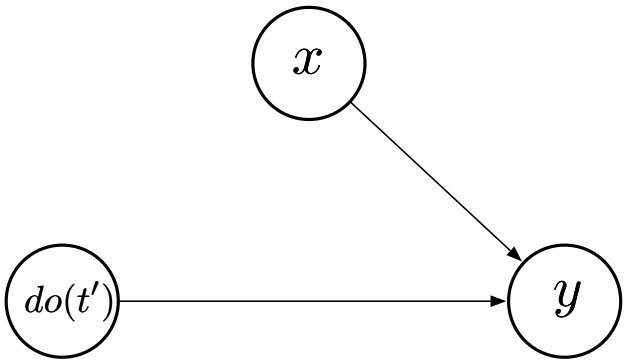
The three directed edges represent the three causal effects:

1. Category of restaurants influences its customer flow. The average customer flow of fast-food restaurants is higher than that of high-end restaurants.
2. Category of restaurants influences its Yelp rating. For example, the average rating of fast-food restaurants is lower than that of high-end restaurants.
3. A restaurant influences its customer flow due to the Yelp rating.



(a)

Figure 2.1 A causal graph used for learning causal effects. Nodes represent random variables and directed edges $x \to y$ indicates that x is a direct cause of y.



(a)

Figure 2.2 A causal graph with intervention. Under intervention $do(t')$, the intervened variable t is fixed to the intervened value t' and all its incoming edges are removed.

Average Treatment Effect. The average treatment effect (ATE) is a measure used to compare treatments (or interventions) in randomized experiments, evaluation of policy interventions, and medical trials. The ATE measures the difference in mean outcomes between units assigned to the treatment and

units assigned to the control. A sample of randomized trials (i.e., an experimental study) can be used to estimate the average treatment effect by comparing the mean outcomes for treated and untreated units. ATEs are typically defined as a parametric measure (i.e., an estimate of the population) that a researcher seeks to know, without reference to the study design or estimation process. It is possible to estimate an ATE in a variety of ways with observational and experimental study designs with random assignment.

Causal Inference in Artificial Intelligence. The causal inference has a close relationship with machine learning. The magnificent progress of deep learning enhances the development of the causal inference area. It proposed an attribution method for neural networks by viewing them as a Structural Causal Model. They also presented that neural attribution can be applied to recurrent neural networks. Experimental results on simulated and real datasets show that the proposed method is promising. A neural structural causal model causalVAE was proposed in [103] to learn the disentanglement features. A Causal Layer is designed to transform independent exogenous factors into causal endogenous factors that represent the causal concepts in data.

Causal search methods are statistical estimation of parameters describing a graphical causal structure. Traditionally, there have been a number of different approaches to causal discovery. The gold standard of causal discovery has typically been to per-

form planned or randomized experiments. Recently, [96, 119] utilized reinforcement learning to help to discover causal structure.

2.2 Reinforcement Learning

Reinforcement learning(RL) is about an agent interacting with the environment. By trial and error, the most important aspect of the real problem facing a learning agent interacting with its environment to achieve a goal. The optimal policy is learned for sequential decision-making problems in different fields such as natural, social sciences, and engineering. Benefiting from big data, powerful computation, i.e.Graphics Processing Unit(GPU), and advanced software and hardware co-design, we have been witnessing the renaissance of reinforcement learning.

Reinforcement Learning & Supervised Learning. In contrast with reinforcement learning, supervised learning relies on training sets of labeled examples provided by a knowledgeable supervisor. Each example is a description of a situation together with a specification: the label of the correct action the system should take in that situation, which is often to identify a category to which the situation belongs. A system that learns this way needs to generalize its responses in order to act correctly in situations outside of the training set. This kind of learning is not adequate for learning from interaction.

Reinforcement Learning & Unupervised Learning. Reinforcement learning is also different from what machine learning researchers call unsupervised learning, which is typically about finding structure hidden in collections of unlabeled data. The fact that reinforcement learning does not rely on examples of correct behavior might lead one to think that it is unsupervised learning. Rather than looking for a hidden structure, reinforcement learning tries to maximize a reward signal.

The Markov Property. The formulation in the simplest possible forms without trivializing is intended to include these three aspects-sensation, action, and goal. At time step t, the agent receives a state s_t in a state space S. An action a_t is selected from an action space A with a policy $\pi(a_t|s_t)$. Normally, $\pi(a_t|s_t)$ is the agent's behavior, i.e., a mapping from state s_t to actions a_t according to the reward function $R(s, a)$. The agent receives a scalar reward r_t, and transitions to the next state s_{t+1} with the state transition probability $P(s_{t+1}|s_t, a_t)$. In an episodic scenario, the agent will continue until reach a terminal state then restarts. Consider a general environment respond at time $t + 1$ to the action taken at time t. In the most general, causal case this response may depend on everything that has happened earlier. In this case the dynamics can be defined only by specifying the complete probability distribution:

$$Pr\{R_{t+1} = r, S_{t+1} = s'|S_0, A_0, R1, ..., S_{t-1}, A_{t-1}, Rt, S_t, A_t\}, \tag{2.1}$$

for all r, s' and all possible values of past events. If the state signal follows the markov property, the environment's response at $t+1$ depends only on the state and action representation at time t. In this case, the dynamics can be defined only by specifying only:

$$p(s', r|s, a) = Pr\{R_{t+1} = r, S_{t+1} = s'|S_t, A_t\}, \qquad (2.2)$$

for all r, s', S_t and A_t. A state signal has the markov property, and is a markov state, if and only if Equation (2.2) is equal to Equation (2.1) for all s', r and histories, $S_0, A_0, R1, ..., S_{t-1}, A_{t-1}, Rt, S_t, A_t$. If an environment has the Markov property, then its one-step dynamics Equation (2.2) enable us to predict the next state and expected next reward given the current state and action. One can show that, by iterating this equation, taking only the current state and a complete history up to the current time, one can predict all future states and expected rewards. It also follows that Markov states provide the best possible basis for choosing actions. This means that the best policy for picking actions as a function of a Markov state is no better than the best policy for choosing actions as a function of complete histories.

Reinforcement Learning in Artificial Intelligence. Deep reinforcement learning gets attention in recent years owing to the blooming of deep learning. The components of reinforcement learning such as value function, $\hat{v}(s;\theta)$ or $\hat{q}(s,a;\theta)$, policy $\pi(a|s;\theta)$, and state transition function and reward function. Here, the parameters θ are the weights of the deep learning

model. The difference between deep RL and conventional RL is that stochastic gradient descent is used as a function approximator to update weight parameters in deep RL. Recent outstanding works like Deep Q-Network and AlphaGo got state-of-the-art results with stabilized training schedules.

□ **End of chapter.**

Chapter 3

General Object Detection

3.1 General Object Detection

3.1.1 Introduction

Object detection is an important task in computer vision, which requires predicting a bounding box of an object with a category label for each instance in an image. State-of-the-art techniques can be divided into either anchor-based methods [4, 28, 34, 35, 37, 57, 69, 70, 72] and anchor-free methods [24, 44, 68, 88, 104, 117]. Recently, the anchor-free methods have increasing popularity over the anchor-based methods in many applications and benchmarks [19, 25, 30, 55]. Despite the success of anchor-free methods, one should note that these methods still have limitations on their accuracy, which are bounded by the way that the bounding boxes are learned in an atomic fashion. Here, we discuss two concerns of existing anchor-free methods which lead to the inaccurate detection.

First, the definition of center key-points [24] is inconsistent with their semantics. As we all know that center key-point is essential for anchor-free detectors. It is a common strategy to embed positive center key-points inside an object bounding box into a Uniform or Gaussian distribution in the training stage of the anchor-free detectors such as FCOS [88] and CornerNet [50]. However, it is inevitable to falsely consider noisy pixels from background as positives, as illustrated in Fig. 3.1. Namely, exploiting a trivial strategy to define positive targets would lead to a significant semantic inconsistency, degrading the regression accuracy of detectors.

Second, local wise regression is limited. Concretely, a center key-point usually provides box predictions in a regional/local-wise manner, which potentially defects the detection accuracy. The local-wise prediction results from the limitation of the receptive fields of convolution kernels, and the design of treating each box prediction from each center key-point as an atomic operation. As shown in Fig. 3.2, the dotted predicted box and corresponding center key-point are presented in the same color. Although each predicted box is surrounding the object, it is imperfect because four boundaries are not well aligned to the ground truth simultaneously. As a result, choosing a box of high score at inference stage as the final detection result is sometimes inferior.

To tackle the inaccurate detection problem, we present a novel bounding box reorganization method, which dives deeper

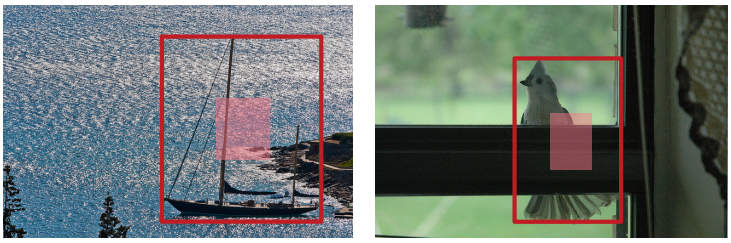


Figure 3.1 **An illustration of the inconsistency between the semantics of center key-points inside a bounding box and their annotations.** Pixels of backgrounds in the red central area are considered as positive center key-points, which is incorrect.

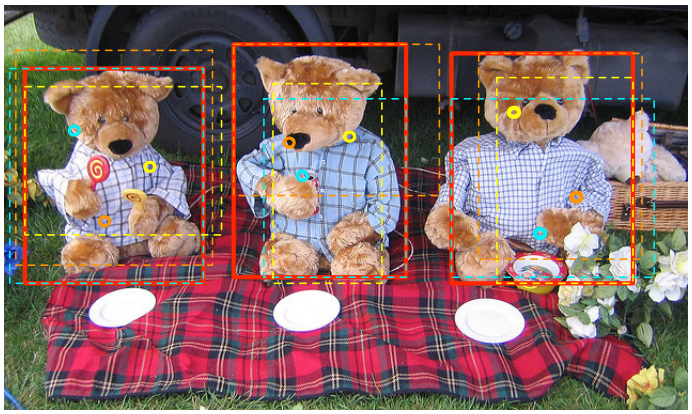


Figure 3.2 **An illustration of the boundary drifts in box predictions of general anchor-free detectors.** Limited by regional receptive fields and the design of treating each box prediction as an atomic operation in general detectors, each predicted box with dotted line is imperfect where four boundaries are not well aligned to the ground truth simultaneously. After box decomposition and combination, the reorganized box with red color gets better localization.

into box regressions of center key-points and takes care of semantic consistencies of center key-points. This reorganization method contains two modules, denoted as box decomposition and recombination (D&R) module and semantic consistency

module. Specifically, box predictions of center key-points inside an instance form an initial coarse distribution of the instance localization. This distribution is not well aligned to the ideal instance localization, and boundary drifts usually occur. The D&R module is proposed to firstly decompose these box predictions into four sets of boundaries to model an instance localization at a lower refined level, where the confidence of each boundary is evaluated according to the deviation with ground-truth. Next, these boundaries are sorted and recombined to form a sort of more accurate box predictions for each instance, as described in Fig. 3.2. Then, these refined box predictions contribute to the final evaluation of box regressions.

Meanwhile, the semantic consistency module is proposed to rule out noisy center key-points coming from the background, which allows our method to focus on key-points that are strongly related to the target instance semantically. Thus, box predictions from these semantic consistent key-points can form a more tight and robust distribution of the instance localization, which further boosts the performance of the D&R module. Our semantic consistency module is an adaptive strategy without extra hyper-parameters for predefined spatial constraints, which is superior to existing predefined strategies in [88, 95, 118].

The main contribution of this work lies in the following aspects.

- We propose a novel box reorganization method in a uni-

fied anchor free detection framework. Especially, a D&R module is proposed to take the boundary prediction as an atomic operation, and then reorganize well-aligned boundaries into boxes in a bottom-up fashion with negligible computation overhead. To the best of our knowledge, the idea of breaking boxes into boundaries for training has never been investigated in this task.

- We evaluate the semantic inconsistency between center keypoints inside an instance and the annotated labels, which helps boost the convergence of a detection network.
- The proposed method DDBNet obtains a state-of-the-art result of 45.5% in AP. The stable experimental results in all metrics ensure that this method can be effectively extended to typical anchor free detectors.

3.1.2 Related Work

Anchor based Object Detectors. In anchor-based detectors, the anchor boxes can be viewed as pre-defined sliding windows or proposals, which are classified as positive or negative samples, with an extra offsets regression to refine the prediction of bounding boxes. The design of anchor boxes is popularized by two-stage approaches such as Faster R-CNN in its RPNs [72], and single-stage approaches such as SSD [57], RetinaNet [54], and YOLO9000 [69], which has become the convention in a modern detector. Anchor boxes make the best use of the feature maps of

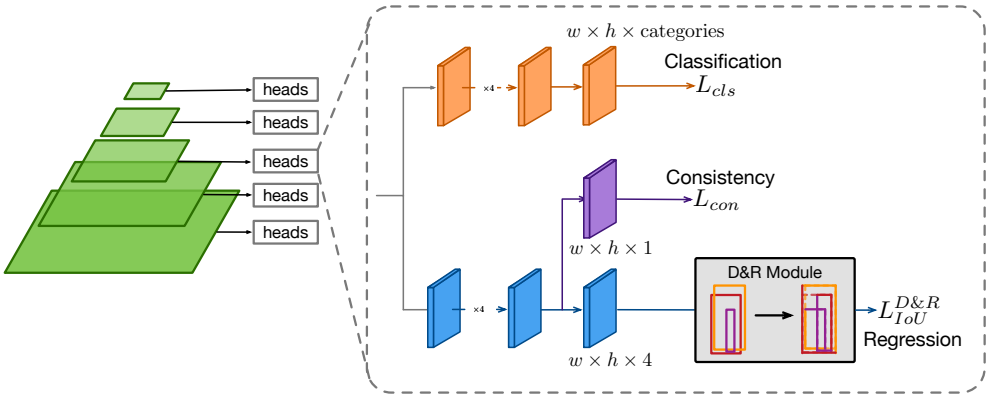


Figure 3.3 **An illustration of our network architecture.** Two novel components: the D&R module and the consistency module are incorporated into a general detection network. The D&R module carries out box decomposition and recombination in the training stage regularized by the IoU loss and predicts boundary confidences supervised by the boundary deviation. The consistency module selects meaningful pixels whose semantics is consistent with the instance to improve network convergence in the training stage.

CNNs and avoid repeated feature computation, speeding up the detection process dramatically. However, anchor boxes result in excessively too many hyper-parameters that are used to describe anchor shapes or to label each anchor box as a positive, ignored or negative sample. These hyper-parameters have shown a great impact on the final accuracy, and require heuristic tuning.

Anchor Free Object Detectors. Anchor-free detectors directly learn the object existing possibility and the bounding box coordinates without anchor reference. DenseBox [41] is a pioneer work of anchor-free based detectors. While due to the difficulty of handling overlapping situations, it is not suitable for generic object detection.

One successful family of anchor free works [48, 88, 95, 118] adopts the Feature Pyramid network [52] (FPN) as the backbone network and applies direct regression and classification on multi-scale features. These methods treat the bounding box prediction as an atomic task without any further investigations, which bounds the detection accuracy due to the two concerns we discussed in the introduction. To avoid the drawback of anchors and refine the box presentations, points based box representation becomes popular recently [24, 50, 104, 117]. For example, CornerNet [50] predicts the heatmap of corners and apply an embedding method to group a pair of corners that belong to the same object. [117] presents a bottom-up detection framework inspired by the keypoint estimations. Compared to these points based methods, our proposed method has following innovations: 1) Our method focuses on the mid-level boundary representations to achieve a balance between accuracy and robustness of feature modeling; 2) Our method does not need to learn an embedding explicitly while obtaining a reliable boundary grouping to produce the final bounding box predictions.

Furthermore, it is observed that anchor-free methods may produce a number of low-quality predicted bounding boxes at locations that are far from the center of a target object. In order to suppress these low-quality detections, a novel "centerness" branch to predict the deviation of a pixel to the center of its corresponding bounding box is exploited in FCOS [88]. This score is then used to down-weight low-quality detected bounding

boxes and merge the detection results in NMS. FoveaBox [48] focuses on the object's center motivated by the fovea of human eyes. It is reasonable to degrade the importance of pixels close to boundaries, but the predefined center field may not cover all cases in the real world, as shown in Fig. 3.1. Thus, we propose an adaptive consistency module to solve the inconsistency issue mentioned above between the semantics of pixels inside an instance and the predefined labels or scores.

3.1.3 Proposed Method

In this work, we build DDBNet based on FCOS as a demonstration, which is an advanced anchor-free method. As shown in Fig. 3.3, our innovations lie in the box decomposition and recombination (D&R) module and the semantic consistency module.

To be specific, the D&R module reorganizes the predicted boxes by breaking them into boundaries for training which is concatenated behind the regression branch. In the training stage, once bounding box predictions are regressed at each pixel, the D&R module decomposes each bounding box into four directional boundaries. Then, boundaries of the same kind are ranked by their actual boundary deviations from the ground-truth. Consequently, by recombining ranked boundaries, more accurate box predictions are expected, which are then optimized by the IoU loss [111].

As for the semantic consistency module, a new branch of estimating semantic consistency instead of centerness is incorporated into the framework, which is optimized under the supervision of the semantic consistency module. This module exploits an adaptive filtering strategy based on the outputs of the classification and the regression branches. More details about the two modules are provided in the following subsections.

Box Decomposition and Recombination

Given an instance I, every pixel i inside of I regresses a box $p_i = \{l_i, t_i, r_i, b_i\}$. The set of predicted boxes is denoted as $B_I = \{p_0, p_1, \ldots, p_n\}$, where l, t, r, b denote the left, the top, the right, and the bottom boundaries respectively.

Normally, an IoU regression loss is expressed as

$$L_{IoU} = -\frac{1}{N_{pos}} \sum_{I} \sum_{i}^{N} \log(IoU(p_i, p_I^*)), \tag{3.1}$$

where N_{pos} is the number of positive pixels of all instances, p_I^* is the regression target. Simply, the proposed box decomposition and recombination (D&R) module is designed to reproduce more accurate p_i with the optimization of IoU loss. As shown in Fig. 3.4, the D&R module consists of four steps before regularizing the final box predictions based on the IoU regression. More details are described as follows.

Decomposition: A predicted box p_i is splited into boundaries l_i, t_i, r_i, b_i and the IoU s_i between p_i and p_I^* is assigned as the

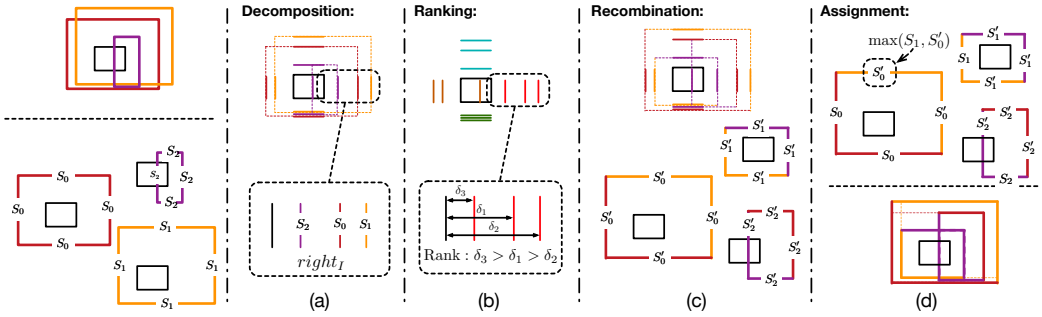


Figure 3.4 **An illustration of the work flow of the D&R module.** For a clear visualization, only three predictions in color are provided for the same ground-truth shown in black. (a) **Decomposition:** Break up boxes and assign IoU scores S_0, S_1, S_2 of boxes to boundaries as confidence. (b) **Ranking:** The rule how we recombine boundaries to new boxes. (c) **Recombination:** Regroup boundaries as new boxes and assign new IoU scores S'_0, S'_1, S'_2 to boundaries as confidence. (d) **Assignment:** Choose the winner confidence as final result. The recombined box is shown on the right.

confidences of four boundaries, as shown in Fig. 3.4(a). For instance I, the confidences of boundaries is denoted as a $N \times 4$ matrix S_I. Then we group four kinds of boundaries into four sets, which are $left_I = \{l_0, l_1, ..., l_n\}$, $right_I = \{r_0, r_1, ..., r_n\}$, $bottom_I = \{b_0, b_1, ..., b_n\}$, $top_I = \{t_0, t_1, ..., t_n\}$.

Ranking: Considering the constraint of the IoU loss [111], where the larger intersection area of prediction boxes with smaller union area is favored, the optimal box prediction is expected to have the lowest IoU loss. Thus, traversing all the boundaries of the instance I to obtain the optimal box rearrangement B'_I is an intuitive choice. However, in this way, the computation complexity is quite expensive, which is $\mathcal{O}(n^4)$. To avoid the heavy computation brought by such brute force method, we ap-

ply a simple and efficient ranking strategy. For each boundary set of instance I, the deviations δ_I^l, δ_I^r, δ_I^b, δ_I^t to the targets boundary $p_I^* = \{l_I, r_I, b_I, t_I\}$ are calculated. Then, boundaries in each set are sorted by the corresponding deviations, as shown in Fig. 3.4(b). The boundary closer to the ground-truth has the higher rank than the boundary farther. We find that this ranking strategy works well and the ranking noise does not affect the stability of the network training.

Recombination: As shown in Fig. 3.4(c), boundaries of four sets with the same rank are recombined as a new box $B'_I = \{p'_0, p'_1, \ldots, p'_n\}$. Then the IoU s'_i between p'_i and p_I^* is assigned as the recombination confidence of four boundaries. The confidences of recombination boundaries is expressed as matrix S'_I with shape $N \times 4$.

Assignment: Now we get two sets of boundaries scores S_I and S'_I. As described as Fig. 3.4(d), the final confidence of each boundary is assigned using the higher score within S_I and S'_I instead of totally using S'_I. This assignment strategy results from the following case, *e.g.* the recombined low-rank box contains boundaries far away from the ground-truth. Then, the confidences s'_i of four boundaries after recombination are much lower than their original one s_i. The severely drifted confidence scores lead to unstable gradient back-propagation in the training stage.

Thus, for reliable network training, each boundary is optimized under the supervision of the IoU loss estimated based

on the ground-truth and the optimal box with its corresponding better boundary score. Especially, our final regression loss consists of two parts:

$$L_{IoU}^{D\&R} = \frac{1}{N_{pos}} \sum_{I} (\mathbb{1}_{\{S'_I > S_I\}} L_{IoU}(B'_I, T_I) + \mathbb{1}_{\{S_I \geqslant S'_I\}} L_{IoU}(B_I, T_I)), \tag{3.2}$$

where $\mathbb{1}_{\{S_I \geqslant S'_I\}}$ is an indicator function, being 1 if the original score is greater than the recombined one, vice versa for $\mathbb{1}_{\{S'_I > S_I\}}$. The gradient of each boundary is selected to update network according to the higher IoU score between the original box and the recombined box. Compared to the original IoU loss Equation (3.1) where gradients are back-propagated in local receptive fields, Equation (3.2) updates the network in context without extra parameterized computations. As box in B'_I is composed by boundaries from different boxes, features are updated in an instance-wise fashion. Note that there are no further parameters added in D&R module. In short, we only change the way how gradient be updated.

Semantic Consistency Module

Since the performance of our D&R module to some extent depends on the box predictions of dense pixels inside an instance, an adaptive filtering method is required to help the network learning focus on positive pixels while rule out negative pixels. Namely, the labeling space of pixels inside an instance is expected to be consistent with their semantics. Different from

previous works [48, 88, 95] which pre-define pixels around the center of the bounding box of an instance as the positive, our network evolves to learn the accurate labeling space without extra spatial assumptions in the training stage.

The formula of semantic consistency is expressed as:

$$\begin{cases} \overline{C_{I\downarrow}} \bigcap \overline{R_{I\downarrow}} \leftarrow \text{negative}, \\ \overline{C_{I\uparrow}} \bigcup \overline{R_{I\uparrow}} \leftarrow \text{positive}, \\ c_i = \max_{j=0}^{g}(c_j) \in C_I, \end{cases} \tag{3.3}$$

where R_I is the set of IoU scores between the ground-truth and the predicted boxes of pixels inside the instance I, $\overline{R_I}$ is the mean IoU score of the set R_I, $\overline{R_{I\downarrow}}$ denotes pixels which have lower IoU confidence than the mean IoU $\overline{R_I}$. Inversely, $\overline{R_{I\uparrow}}$ denotes pixels which have higher IoU confidence than $\overline{R_I}$. The element $c_i \in C_I$ is the maximal classification score among all categories of the i-th pixel, and g denotes the number of categories. Similarly, $\overline{C_{I\downarrow}}$ denotes pixels which have lower classification scores than the mean score of C_I. Labels of categories are agnostic in this approach so that the predictions of incorrect categories will not be rejected during training. Finally, as shown in Fig. 3.5, the intersection pixels in $\overline{R_{I\downarrow}}$ and $\overline{C_{I\downarrow}}$ are assigned negative, while the union pixels in $\overline{R_{I\uparrow}}$ and $\overline{C_{I\uparrow}}$ are assigned positive. Meanwhile, if pixels are covered by multiple instances, they prefer to represent the smallest instance.

More to the point, the filtering method determined by Equa-

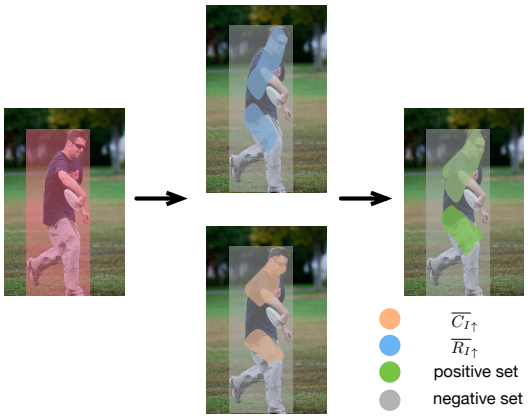


Figure 3.5 **Visualized example of semantic consistency module.** The intersection regions of positive regression and positive classification sets are regarded as consistent targets.

tion (3.3) is able to adaptively control the ratio of positive and negative pixels of instances with different sizes during the training stage, which have a significantly effect on the detection capability of the network. In the experiments, we investigate the performance of different fixed ratio, and then find that the adaptive selection by mean threshold performs best.

After the labels of pixels are determined autonomously according to the semantic consistency, the inner significance of each positive pixel is considered in the learning process of our network, similarly to the centerness score in FCOS [88]. Thus, our network is able to emphasize on more important part of an instance and is learnt more effectively. Especially, the inner significance of each pixel is defined as the IoU between the predicted box and the ground-truth. Then, an extra branch of estimating the semantic consistency of each pixel is added to the

network supervised by the inner significance. The loss for semantic consistency is expressed as in Equation (3.4), where r_i is the output of semantic consistency branch. $IoU(p_i, p_I^*)$ denotes the inner significance of each pixel.

$$L_{con} = \frac{1}{N_{pos}} \sum_{I} \sum_{i \in \overline{C_I}_{\uparrow} \bigcup \overline{R_I}_{\uparrow}} CE(r_i, IoU(p_i, p_I^*)). \tag{3.4}$$

Generally, the overall training loss is defined as:

$$L = L_{cls} + L_{reg}^{D\&R} + L_{con}, \tag{3.5}$$

where L_{cls} is the focal loss as in [54].

3.2 Experiments

3.2.1 Experimental Setting

Dataset. Our method is comprehensively evaluated on a challenging COCO detection benchmark [55]. Following the common practice of previous works [50, 54, 88], the COCO *trainval35k* split (115K images) and the *minival* split (5K images) are used for training and validation respectively in our ablation studies. The overall performance of our detector is reported on the *test-dev* split and is evaluated by the server.

Network Architecture. As shown in Fig. 3.3, Feature Pyramid Network (FPN) [52] is exploited as the fundamental detection network in our approach. The pyramid is constructed with the levels $P_l, l = 3, 4, ..., 7$ in this work. Note that each pyramid

level has the same number of channels (C), where $C = 256$. At the level P_l, the resolution of features is down-sampled by 2^l compared to the input size. Please refer to [52] for more details. Note that four heads are attached to each layer of FPN. Apart from the regression and classification heads, a head for semantic consistency estimation is provided, consisting of a normal convolutional layer. The regression targets of different layers are assigned in the same way as in [88].

Training Details. Unless specified, all ablation studies take ResNet-50 as the backbone network. To be specific, the stochastic gradient descent (SGD) optimizer is applied and our network is trained for 12 epochs over 4 GPUs with a minibatch of 16 images (4 images per GPU). Weight decay and momentum are set as 0.0001 and 0.9 respectively. The learning rate starts at 0.01 and reduces by the factor of 10 at the epoch of 8 and 11 respectively. Note that the ImageNet pre-trained model is applied for the network initialization. For newly added layers, we follow the same initialization method as in RetinaNet [54]. The input images are resized to the scale of 1333×800 as the common convention. For comparison with state-of-the-art detectors, we follow the setting in [88] that the shorter side of images in the range from 640 to 800 are randomly scaled and the training epochs are doubled to 24 with the same reduction at epoch 16 and 22.

Inference Details. At post-processing stage, the input size of images are the same as the one in training. The predictions with

Table 3.1 **Comparison with state-of-the-art two stage and one stage Detectors** (*single-model and single-scale results*). DDBNet outperforms the anchor-based detector [54] by 2.9% AP with the same backbone. Compared with anchor-free models, DDBNet is in on-par with these state-of-the-art detectors. † means the NMS threshold is 0.6 and others are 0.5.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods:							
Faster R-CNN w/ FPN [52]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN w/ TDM [79]	Inception-ResNet-v2-TDM [82]	36.8	57.7	39.2	16.2	39.8	52.1
Faster R-CNN by G-RMI [40]	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
RPDet [104]	ResNet-101-DCN	42.8	65.0	46.3	24.9	46.2	54.7
Cascade R-CNN [4]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
One-stage methods:							
YOLOv2 [69]	DarkNet-19 [69]	21.6	44.0	19.2	5.0	22.4	35.5
SSD [57]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD [28]	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
FSAF [118]	ResNet-101	40.9	61.5	44.0	24.0	44.2	51.3
RetinaNet [54]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	53.9
CornerNet [50]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
ExtremeNet [117]	Hourglass-104	40.1	55.3	43.2	20.3	43.2	53.1
FCOS† [88]	ResNet-101-FPN	41.5	60.7	45.0	24.4	44.8	51.6
FCOS† [88]	ResNeXt-64x4d-101-FPN	43.2	62.8	46.6	26.5	46.2	53.3
FCOS† w/improvements [88]	ResNeXt-64x4d-101-FPN	44.7	64.1	48.4	27.6	47.5	55.6
DDBNet (Ours)	ResNet-101-FPN	42.0	61.0	45.1	24.2	45.0	53.3
DDBNet (Ours)	ResNeXt-64x4d-101-FPN	43.9	63.1	46.7	26.3	46.5	55.1
DDBNet (Ours)§	ResNeXt-64x4d-101-FPN	**45.5**	**64.5**	**48.5**	**27.8**	**47.7**	**57.1**

§ GIoU [73] and Normalization methods of 'improvements' proposed in FCOS [88] are applied, ctr.sampling in 'improvements' [88] are not compatible with our setting and we do not use.

classification scores $s > 0.05$ are selected for evaluation. With the same backbone settings, the inference speed of DDBNet is same as the detector in FCOS [88].

3.2.2 Overall Performance

We compare our model denoted as DDBNet with other state-of-the-art object detectors on the *test-dev* split of COCO bench-

mark, as listed in TABLE 3.1. Compared to the anchor-based detectors, our DDBNet shows its competitive detection capabilities. Especially, it outperforms RetinaNet [54] by 2.9% AP. When it comes to the anchor-free detectors, especially detectors such as FCOS [88] and CornerNet [50] benefiting from the point-based representations, our DDBNet achieves performances gains of 0.5% AP and 1.5% AP respectively. Based on the ResNeXt-64x4d-101-FPN backbone [98], DDBNet works better than [88] with a 0.7% AP gain. Especially for large objects, our DDBNet gets 55.1% AP, better than 53.3 % reported in FCOS [88]. We also apply part of 'improvement' methods proposed in FCOS to DDBNet and gets 0.8% better than the FCOS with all 'improvements' applied. To sum up, compared to detectors exploiting point-based representations, our DDBNet can similarly benefit from the mid-level boundary representations without heavy computation burdens. Furthermore, DDBNet is compared to several two stage models. It overpasses [52] by a large margin.

3.2.3 Ablation Study

In this section, we explore the effectiveness of our method, including two main modules of box D&R module and semantic consistency module. Additionally, we conduct in-depth analysis of the performance metrics of our method.

Comparison with Baseline Detector

It should be noted that FCOS detector [88] without the centerness branch in both training and inference stages is taken as our baseline. Here we conduct in-depth analysis of the performance metrics of our method.

Box D&R module. As shown in TABLE 3.2, by incorporating the D&R module into the baseline detector, a 1.2% AP gain is obtained, which proves that our D&R module can boost the overall performance of the detector. Especially for the AP_{75}, a 1.4% improvement is achieved, which means that D&R performs better on localization even in a strict IOU threshold. Furthermore, D&R module achieves a better performance on large instances according to the large gain on AP_L. With explicit boundary analysis, large instances are often surrounded by numbers of predicted boxes. As a result, it gets easier to find the well-aligned boundaries, then the boxes re-organization can be more effective. Compared to the baseline results in metrics including AP_{50}, AP_S and AP_M, D&R obtains stable performance gains respectively, which shows the stability of our proposed module. By breaking the atomic boxes into boundaries, D&R module makes each boundary find the better optimization direction. The optimization of boundary is not limited by the box its in, instead of depending on a sorted of related boxes. Generally, by adjusting the boundary optimization, the detection network is learnt better.

Semantic Consistency module. The semantic consistent module described in Section 3.1.3 presents an adaptive filter-

Table 3.2 **Ablative experiments for DDBNet on the COCO *minival* split.** We evaluate the improvements brought by the Box Decomposition and Recombination(D&R) module and the semantic consistency module.

Modules			AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	D&R	Consistency						
✓			33.6	53.1	35.0	18.9	38.2	43.7
✓	✓		34.8	54.0	36.4	19.7	39.0	44.9
✓		✓	37.2	55.4	39.5	21.0	41.7	48.6
✓	✓	✓	**38.0**	**56.5**	**40.8**	**21.6**	**42.4**	**50.4**

ing method. It forces our detection network into autonomously focusing on positive pixels whose semantics are consistent with the target instance. As shown in TABLE 3.2, the semantic consistency module contributes to a significant performance gain of 3.6% AP compared to the baseline detector. This variant surpasses the baseline by large margins in all metrics. Due to that the coarse bounding boxes would contain backgrounds and distractors inevitably, the network is learnt with less confusion about the targets when equips our adaptive filtering module. More ablation analysis on semantic consistency module is provided in Section 3.2.3.

Cooperation makes better. In our final model denoted as DDBNet, the semantic consistency module first filters out a labeling space of pixels inside each instance that is strongly relative to the geometric and semantic characteristics of the instance. The box predictions of the filtered positive pixels are further optimized by the D&R module, leading to more accurate detection results. Consequently, DDBNet achieves 38% AP,

better than all the variants in TABLE 3.2. Our method boosts detection performance over the baseline by 2.7%, 4.2%, and 6.7% respectively on AP_S, AP_M, AP_L.

Analysis on D&R Module.

Statistical comparison with conventional IoU Loss. As we mentioned in Section 3.1.3, IoU loss with D&R updates the gradient according to the optimal boundary scores. To confirm the stability of D&R module, we plot the average IoU scores and variances of boxes before and after D&R respectively. We can see that with D&R module, the average values of IoU scores are higher than the means of origin IoU scores by a large margin around 10% in the whole training schedule, as in Fig. 3.6. At the start of training, the mean of optimal boxes gets 0.47 which is better than 0.34 of origin boxes. As training goes on, both average scores of origin and optimal boxes increase and remain at 0.77 and 0.86 at the end. Variances of IoU scores with D&R are much lower than the origin IoU scores, which indicates D&R module improves the overall quality of boxes and provides better guidance for training.

Visualization on D&R module. We provide some qualitative results of box predictions before and after incorporating the D&R module into the baseline detector, as shown in Fig. 3.7. For clear visualization, we plot origin boxes and boxes after recombination individually. Predictions are presented in green

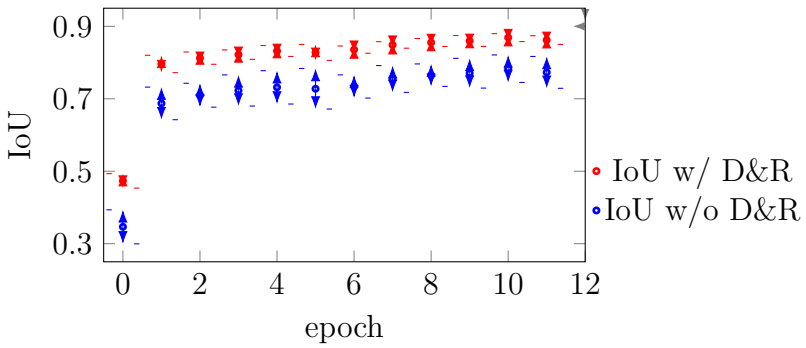


Figure 3.6 **Average IoU scores for all predicted boxes during the training.** The red points denote the IoU scores with D&R module while the blue points are the IoU scores without optimization. Vertical lines indicate the variance of IoU scores.

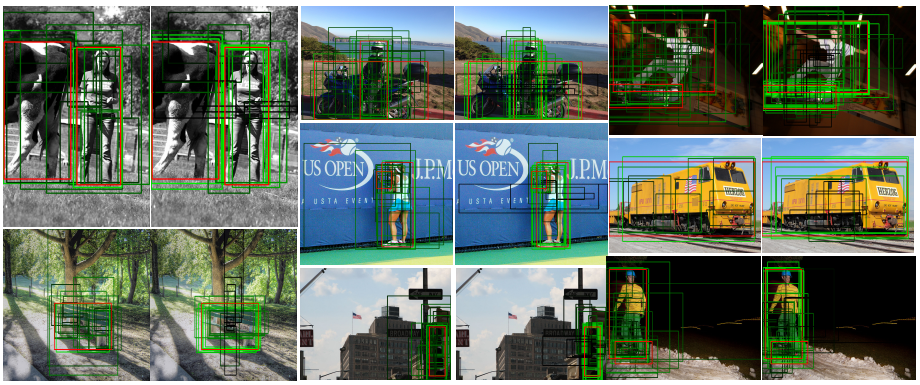


Figure 3.7 **Illustration of improved box predictions provided by our DDBNet.** We visualize the boxes before the decomposition (left images of the pairs) and the boxes after the recombination (right images of the pairs). **Red**: ground-truth boxes. **Green**: the predictions, where the lighter colors indicate higher IoU scores. **Black**: the boxes with low score, which will be masked according to the regression loss. Boxes ranked by D&R module are much better organized than the origin boxes and the localizations are much correlated to the instances. All the results are from DDBNet with ResNet-50 as backbone on *trainval*35*k* split.

Table 3.3 **Comparison among different positive assignment strategies.** 'None' means no sampling method is applied. 'PN' denotes as the definition in [48], which means center regions are positive and others are negative. 'PNI' is the sampling used in [95,118], ignore regions are added between positive and negative. Note that the consistency term is not included in this table.

Settings	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
None	33.6	53.1	35.0	18.9	38.2	43.7
PN	34.2	53.2	36.3	20.8	38.9	44.2
PNI	33.7	53.0	35.5	17.9	38.3	44.1
Ours	**35.3**	**55.4**	**37.1**	**20.9**	**39.6**	**45.9**

and the lighter colors indicate higher IoU scores. With D&R module, boundaries are recombined together to obtain a tighter box of each instance. The distribution of boxes after D&R module is fitter than the origin boxes which is robust than the conventional regression. As we mentioned in Section 3.1.3, there exists recombined low-rank boxes with boundary scores lower than the origin. These boundaries are masked according to the Equation (3.2).

Analysis on Semantic Consistency

Dynamic or predefined positive assignment. To further show the superiority of dynamic positive assignment in semantic consistency module, we investigate other variants using different predefined strategies mentioned in previous works. Fove-

aBox [48] (denoted as 'PN') applies center sampling in their experiments to improve the detection performance. This center sampling method defines the central area of a target box based on a constant ratio as positive while the others as negative. 'PNI' is taken used in [95, 118] which exploits positive, ignore and negative regions for supervised network training. According to the result in TABLE 3.3, 'PN' (second line) gets slight improvement compared to the baseline where no sampling method is adopted. So restricting the searching space to the central area makes sense in certain cases and indeed helps improve object detection. But the 'PNI' gets a lower performance, especially on AP_S. Namely, adding an ignore region between the ring of negative areas and the central positive areas does not further improve the performance and gets a large drop on the detection of small objects. The limited number of candidates of small objects and the lower ratio of positive candidates in 'PNI' result in the poor detection capability. Contrastively, our proposed filtering method does not need to pre-define the spatial constraint while show best performances in all metrics.

Adaptive or constant ratio. As mentioned in section 3.1.3, we investigate the constant ratio to replace the adaptive selection by mean. Four variants are obtained where the constant ratio is set from 0.4 to 0.7. For instance I with M candidates, top $\lfloor c \times M \rfloor$ candidates are considered as positive, and others are negative, where c is the constant sampling ratio applied to all instances. As shown in TABLE 3.4, these results indicate

Table 3.4 **Comparison among different ratio settings.** where c is the sampling ration for each instance.

ratios	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
$c = 0.4$	34.6	54.2	36.6	19.1	38.5	45.2
$c = 0.5$	34.1	53.5	35.9	19.2	38.4	44.2
$c = 0.6$	34.7	54.2	36.5	19.0	38.7	45.5
$c = 0.7$	35.1	54.6	**37.1**	19.3	39.1	45.7
mean	**35.3**	**55.4**	**37.1**	**20.9**	**39.6**	**45.9**

that the adaptive way in our method is better than the fixed way to select positives from candidates.

□ End of chapter.

Chapter 4

Layout Hotspot Detection

4.1 Introduction

With the development of the semiconductor industry, transistor feature size shrinks rapidly, which significantly challenges manufacturing yield. For instance, low-fidelity pattern on wafer (a.k.a. "hotspot") is one of the emerging issues in the manufacturing [64, 110]. To ensure the printability of layout designs, an efficient and accurate hotspot detector is indispensable. Currently, there are three main classes of methods: lithography simulation, pattern matching and machine learning. By using complicated lithography models to identify hotspots, lithography simulation [46, 75] is accurate but extremely time-consuming. High performance clusters with amounts of nodes are needed in the whole simulation flow and several days are required to complete it. As good replacements of simulation-based methods, pattern matching and machine learning-based techniques are proposed to accelerate the hotspot detection flow while the

detection accuracy is attained as much as possible.

Pattern matching is to set up a collection of hotspot layout patterns to identify any matched patterns in a new design as hotspots [45, 92, 97, 105, 113]. For example, in [113], critical topological features of hotspots are extracted as design rules in design rule checking to locate the hotspot positions. To handle partially variant layout patterns from the pre-defined hotspots, Wen *et al.* [97] proposed a fuzzy matching model which integrates both pattern-matching and machine-learning techniques to dynamically tune the fuzzy region around a known hotspot. Although the pattern matching overcame the runtime issue, this approach, including fuzzy pattern matching, cannot give a confident result on unseen hotspot patterns.

Machine learning-based methods have shown the capability to offer accurate solutions to both known and unknown hotspot patterns with generalized feature extractors [20, 21, 23, 32, 58, 60, 89, 107, 109, 114–116]. A learning model is usually trained by features which are extracted from a batch of labeled data and then conducts hotspot prediction on new layout patterns efficiently. Ding *et al.* [21] exploited a meta-classifier which combines pattern matching and machine learning methods into a unified framework. In [109], a detection flow based on critical feature extraction and PCA-SVM classifier is proposed. To update the learning model with newly detected and verified layout patterns, Zhang *et al.* [115] investigated a classifier based on smooth boosting and optimized concentric circle sampling

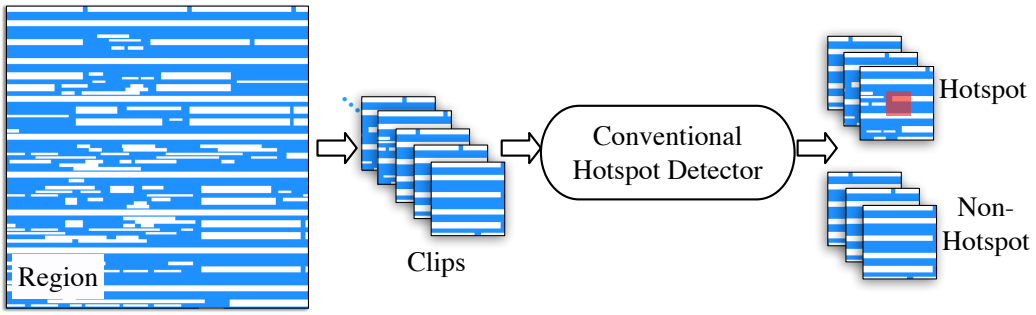


Figure 4.1 The conventional hotspot detection flow.

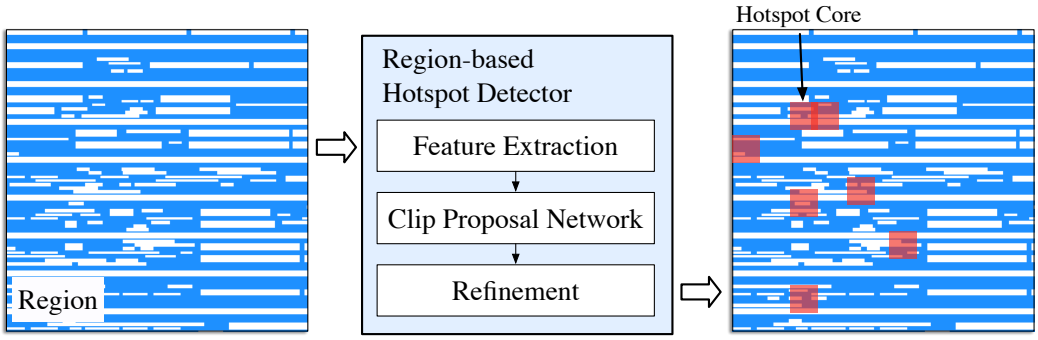


Figure 4.2 The proposed region-based hotspot detection flow.

feature extractor. Recently, Ye *et al.* [107] pointed out the uncertainty problem in hotspot detection and presented a Gaussian process assurance to provide confidence in each prediction. Conventional machine learning approaches achieve good performance, but they are limited to manually-crafted feature extractors. Besides, these approaches are challenged by scalability requirements for printability estimation of a large scale layout.

Convolutional neural networks (CNNs) have become a powerful technique to improve hotspot detection performance [16, 33, 43, 59, 77, 99–102, 108], thanks to its non-linearity and multi-level feature extraction in an automatic manner. For example,

Yang *et al.* [100] investigated a deep CNN which considered the data unbalanced issue and achieved high classification accuracy. Additionally, they designed a biased learning technique for an unbalanced dataset and applied discrete-cosine transformation (DCT) to give proper feature expression [102]. To handle the scenario that labeled data are limited, a semi-supervised neural network is built in [16]. In [43], Jiang *et al.* proposed a binarized neural network to further enhance the performance of the detector.

In literature, however, hotspot detectors only work on small clips extracted from a whole chip layout and can only detect one hotspot location at a time that occurs at a center (i.e. core in [91]) of each clip. Fig. 4.1 illustrates a conventional hotspot detection scheme, which requires repeatedly scanning overlapping regions of a full chip design. Therefore, it could be a waste of computational resources and time-consuming when facing with extremely large layouts. To solve this problem, [12] proposed a new faster region-based hotspot detection framework, which can mark multiple hotspot locations within a region that is much larger than a clip applied in previous works, as shown in Fig. 4.2. To the best of our knowledge, [12] is the first art to design a hotspot detector on detecting multiple processes weak points within very large scale layout clips in one step feed-forward detection. The framework contains a regression and classification multi-task flow which guide to higher accuracy, higher detection speed and lower false alarm penalty. How-

ever, there still exist some defects in our design for region-based hotspot detector. For example, due to the single-scale description of layouts, the encoder-decoder structure limits the feature expression of our extractor. Additionally, the regression loss in [12] handles the coordinates individually where the geometric constraint is not considered. Consequently, multi-branch design for encoder-decoder and IoU regularizer are proposed to enhance the preliminary region-based hotspot detector. The main contributions of this paper are listed as follows:

- We construct a deep neural network specifically for region-based hotspot detection task and our network framework can be efficiently trained end-to-end.
- A clip proposal network and a refinement stage are built to further improve accuracy and reduce false alarm.
- We apply a novel classification and regression strategy to reduce the detection region and make the multiple hotspot detection become realizable in large scales.
- Multi-branch design for encoder-decoder and IoU regularization is introduced to further strengthen the proposed region-based hotspot detector.
- Experimental results show that our proposed framework has great advantages over the state-of-the-art detectors. It can achieve 7.40% accuracy improvement and 13× speedup on average.

The rest of the paper is organized as follows. Section 4.2 introduces basic concepts and gives problem formulation. Section 4.3 discusses the details of the proposed end-to-end neural framework. Section 4.4 introduces the techniques to raise the performance of the proposed detector. Section 4.5 lists experimental comparisons with state-of-the-art methods, followed by conclusion in Section 4.6.

4.2 Preliminaries

Due to the manufacturing process variation, designed layout patterns stochastically cause defects on wafers during the lithographic process. These sensitive patterns may cause reduction of manufacturing yield or even potential circuit failures. Layout patterns that are sensitive to process variations are defined as *hotspots*. We also define a *hotspot clip* as a clip that contains at least one hotspot at its core region [91]. Here the core region is the middle area in the clip. In this paper, the following definitions and metrics are used to evaluate the performance of a hotspot detector.

Definition 1 (Accuracy)**.** *The ratio between the number of correctly detected hotspots and the number of ground truth hotspots.*

Definition 2 (False Alarm)**.** *The number of non-hotspot clips that are detected as hotspots by the classifier.*

Definition 3 (F1 score)**.** *The weighted harmonic mean of the*

test's precision and recall. The score is calculated according to

$$F1 = \left(\frac{2}{recall^{-1} + precision^{-1}} \right). \tag{4.1}$$

It should be noted that the accuracy is also equivalent to the *true positive rate* and the false alarm corresponds to the number of *false positives*. Because a good hotspot detector aims to recognize as many real hotspots as possible and avoids incorrect predictions on non-hotspot patterns, with the evaluation metrics above, we define the region-based hotspot detection (R-HSD) problem as follows.

Problem 1 (R-HSD)**.** *Given a layout region that consists of hotspot and non-hotspot patterns, the objective of region-based hotspot detection is training a model to locate and classify all the hotspot and the non-hotspot within the region, such that the detection accuracy is maximized with minimum false alarm penalty.*

4.3 R-HSD Neural Network

Our proposed region-based hotspot detection (R-HSD) neural network, as illustrated in Fig. 4.2, is composed of three steps: (1) feature extraction, (2) clip proposal network, and (3) refinement. In this section, we will discuss each step with details. At first glance, the R-HSD problem is similar to object detection problem, which is a hot topic in computer vision domain recently. In object detection problem, objects with different

shapes, types and patterns are the instances to be detected. However, as we will discuss, there is a gap between hotspot detection and object detection, e.g. the hotspot pattern features are quite different from the objects in real scenes, thus typical strategies and framework utilized in object detection cannot be applied here directly.

4.3.1 Feature Extraction

Because of wide variations between traditional objects in real scenarios and VLSI layout patterns, it is extremely crucial to design an appropriate feature extractor in our neural network framework. Our feature extractor aims at transforming original layout features non-linearly while reducing computation overhead. Besides, we also tend to enrich the feature diversity with fewer parameters. Based on these two major principles, a feature extractor based on encoder-decoder structure and inception-based modules for efficient extraction is designed. Three convolution layers and two max-pooling layers connect the encoder-decoder structure and inception-based modules. This connection is applied to compress the feature map size from 224×224 to 56×56 which can bring speed-up at the training stage and inference stage.

Yang *et al.* [100] successfully applied DCT to manually extract layout pattern features. Although DCT keeps the spatial information, inevitably, this manual design of the feature ex-

pression ignores some key features thus may not give a comprehensive expression. Furthermore, the processing of the DCT is very time-consuming. Compared to the manually rule-based DCT, our proposed feature extractor can transform the origin layout into a network-compatible expression automatically. As the feature extractor is a part of the whole convolutional neural network, the training procedure is more flexible and efficient.

The convolutional network structure designed by [100] performs well feature extraction, but its structure is too simple thus is limited to the single clip hotspot detection problem. The naive replacement on DCT without redesign on further extractor is not available to the region-based task. There are two metrics for us to think about how to design a new structure in our work. One is going deeper with more layers, while the other is going wider with multiple branches.

Encoder-Decoder Structure

The encoder-decoder structure has been successfully applied to many computer vision tasks, including object detection [29, 53, 79]. The vanilla encoder consists of several convolution layers and the decoder includes the same number of deconvolution layers. The encoder gradually extracts the features from the origin image space to high dimensions latent space by increasing the number of the convolution kernels, then the decoder gradually downsamples the features from high dimensions to origin image space with the symmetrical kernel settings.

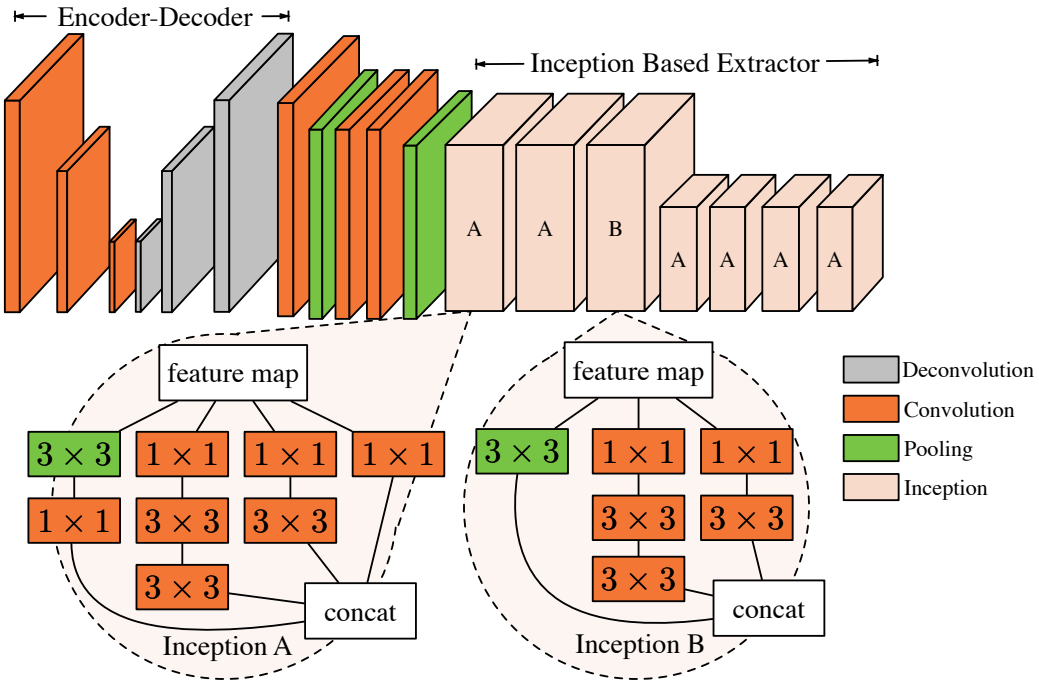


Figure 4.3 The tensor structure of feature extractor.

Convolution Layer. The convolution layer is the major part of the convolution neural network which has been widely used. The operation between tensor and kernel can be expressed as:

$$\mathbf{F} \otimes \mathbf{K}(j,k) = \sum_{i=1}^{c} \sum_{m_0=1}^{m} \sum_{n_0=1}^{m} \mathbf{F}(i, j-m_0, k-n_0)\mathbf{K}(m_0, n_0), \tag{4.2}$$

with tensor $\mathbf{F} \in \mathbb{R}^{c \times p \times p}$ and kernel $\mathbf{K} \in \mathbb{R}^{c \times m \times m}$.

Deconvolution Layer. In contrast to convolutional layers, deconvolution layers do the inverse operation which maps the single input feature point to multiple outputs, which can be considered as a feature generation. The expression can be

written as:

$$\mathbf{T} \otimes \mathbf{K}(j,k) = \sum_{i=1}^{c} \sum_{m_0=1}^{m} \sum_{n_0=1}^{m} \mathbf{T}(i, j-m_0, k-n_0) \mathbf{K}(m-m_0, n-n_0), \tag{4.3}$$

where $\mathbf{T} \in \mathbb{R}^{c \times n \times n}$ is the tensor $\mathbf{F} \in \mathbb{R}^{c \times p \times p}$ padded with zero and $n = (m-1) \times 2 + p$, kernel $\mathbf{K} \in \mathbb{R}^{c \times m \times m}$. Here padding size is the number of pixels we fill on the border of the origin feature maps. In our experiments, the size of a padded feature map is equal to the size of output. Kernel size is the size of the deconvolution kernel. We use 3×3 kernel size which is same as the encoder part. During training, the feature map of the deconvolution layer is updated with the back-propagation.

Inception-based Structure

Empirically, a deeper neural network can give a much more robust feature expression and get higher accuracy compared to a shallow structure as it increases the model complexity. However, deeper networks are prone to overfitting, and gradient vanishing attaches to it. Even worse, it brings sacrifice on speed at both the inference stage and the training stage. Another point we need to take into consideration is that features extracted from the encoder-decoder structure are still in low dimension space. In other words, more convolution kernels are needed. Additionally, salient parts in images (i.e. hotspots in our case) may have pretty large variations in locations and sizes. According to these

issues, we propose an inception-based structure. The following three points are the main rules of our design:

- Increase the number of filters in width at each stage. For each stage, multiple filters do the convolution operation with different convolution kernel size and then concatenate them in channel direction as feature fusion.
- Prune the depth of the output channel for each stage.
- Downsample the feature map size in height and width direction.

With the above rules, the inception structure [83] can take a good balance between the accuracy and the time. The blobs showed in Fig. 4.3 are what we apply in our framework. We construct the module A with the operation stride one and four branches. The aim of module A is to extract multiple features without downsampling the feature map. The operation stride of each layer in module B is two. Here the stride denotes the convolution operation step of kernels on feature maps, the larger strides can decrease the tensor size and reduce the number of operations in subsequent layers. Note that Module B has the same design principles as Module A, the bonus of Module B is to reduce the spatial size of features. We only use one Module B here, because the feature map size should not be too small, while the low dimension of feature expression in final layers may bring negative affects to the final result. The output feature size

of final module A in Fig. 4.3 is 14×14, which is the input feature map of Clip Proposal Network in Fig. 4.4. It can be seen that 1×1 convolution kernel has been applied in both modules. The exploited 1×1 convolution kernel with low channel numbers offers a channel-wise pooling, which brings the dimension reduction by decreasing the number of feature maps whilst retaining their salient features. This technique successfully controls the number of the parameters and convolutional operations, and thus reduces the computational overhead.

In summary, the inception structure brings more abundant feature expressions, which gives the network the ability to do the kernel selection with no operation penalty.

4.3.2 Clip Proposal Network

Given the extracted features, a clip proposal network is developed here to detect potential hotspot clips. For both feature maps and convolutional filters, the tensor structures of the clip proposal network are illustrated in Fig. 4.4. Per preliminary experiments, clips with single aspect ratio and scale (e.g. square equal to the ground truth) may lead to bad performance. Therefore, for each pixel in a feature map, a group of 12 clips with different aspect ratios is generated.

The network is split into two branches: one is for classification and the other is for regression. In classification branch, for each clip, a prediction of hotspot and non-hotspot is calcu-

lated through softmax function. The basic sampling strategy to train the classifier is that the clips highly overlapped with ground-truth are regarded as positive samples and the ones with lower overlaps are considered as negative samples. Apparently, it needs some tweaks and compromises to separate hotspots and non-hotspots. In regression branch, the location and the shape of each clip are determined by a vector $[x, y, w, h]$, where x and y refer to the location of a clip center while w and h respectively record the width and the height of a clip. One criterion for the regressor during training is that clips labeled as non-hotspots are not fed into the regression branch since there are no ground-truth clips for them. The output of our clip proposal network is a bunch of proposals that will be examined by the above-mentioned classifier and regressor to eventually check the occurrence of hotspots. More precisely, it predicts the possibility of a clip being a hotspot or not, and refines the clip.

Clip Pruning

While the number of clips is extremely large during training, high-quality clips should be reserved to train the classifier and the low-quality clips which have medium intersection area to the ground truth should be removed as they are the noises to the classifier. For the clip regression task, it is not reasonable to consider linear regression on these clips with large offset to the ground truth clips. To overcome this problem, we consider automatic clip pruning in our neural network.

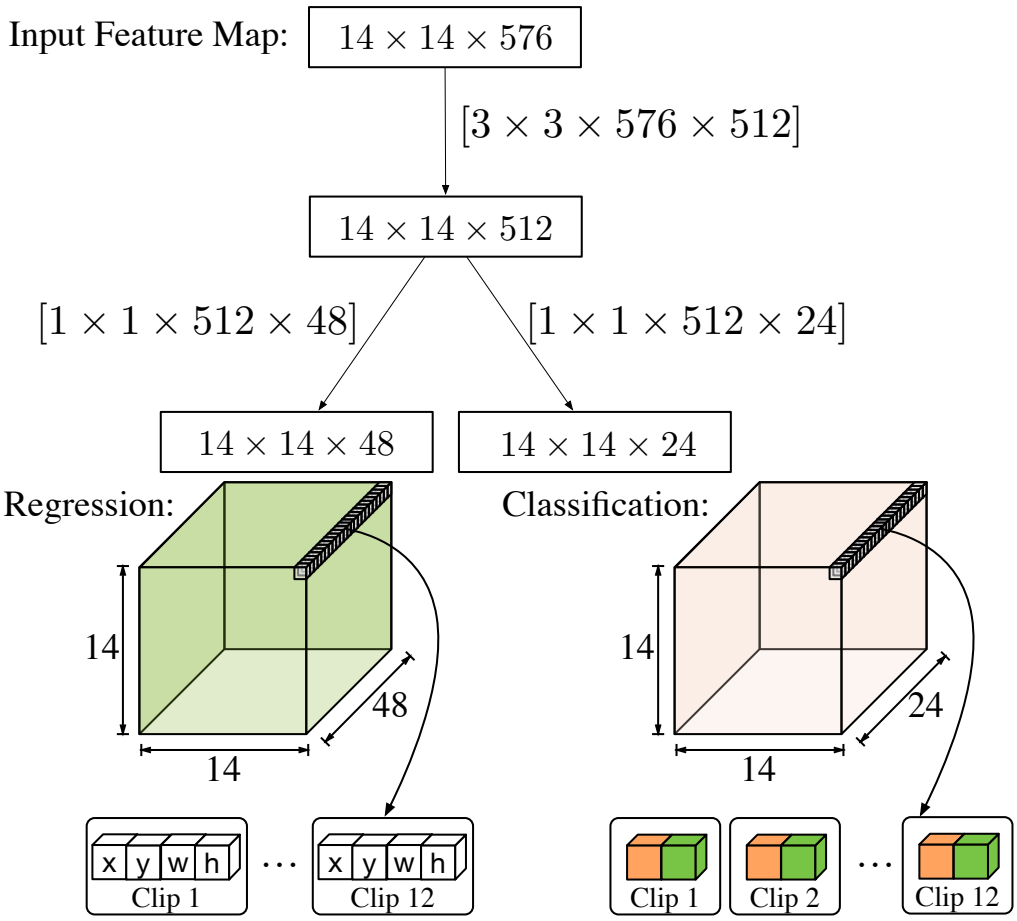


Figure 4.4 The kernel work flow of clip proposal network.

We first define intersection of union (IoU) as follows:

$$\text{IoU} = \frac{clip_{groundtruth} \bigcap clip_{generated}}{clip_{groundtruth} \bigcup clip_{generated}}. \tag{4.4}$$

Then the following clip pruning rules are established:

- A clip's IoU with ground truth clip higher than 0.7 should be reserved as a positive sample;
- The clip's IoU with any ground truth highest score should be reserved as a positive sample;

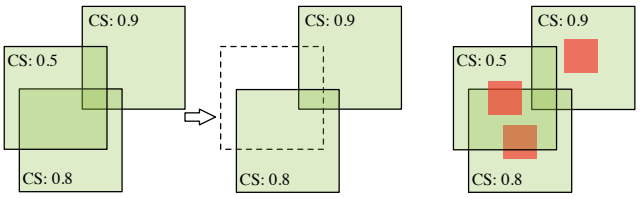
- A clip's IoU with ground truth clip lower than 0.3 should be reserved as a negative sample;
- Rest of clips do no contribution to the network training.

Hotspot Non-Maximum Suppression

After the classification and regression, the distance between some neighbor hotspots is quite close to each other, there exists a set of overlapped clips which have the same regression target. To avoid these redundant calculations at training and inference stages, we develop a hotspot non-maximum suppression (H-NMS) strategy to remove these clips. The H-NMS strategy is shown in Algorithm 1.

Algorithm 1 hotspot non-maximum suppression

```
sorted_ws ← sorted clip set;
k ← size of clip set;
for i ← 1, 2, ..., k do
    current_w ← sorted_ws[i];
    for j ← i, i + 1, ..., k do
        compared_w ← sorted_ws[j];
        Overlap ← Centre_IoU(current_w, compared_w);
        if Overlap > threshold then
            Remove compared_w; k ← k − 1;
        end if
    end for
end for
return sorted_ws;
```



(a) (b)

Figure 4.5 Examples of (a) conventional non-maximum suppression, and (b) the proposed hotspot non-maximum suppression.

The elements of *sorted_ws* are arranged in descending order according to the classification score (line 1). `Centre_IoU` is a function returning the IoU score which focus on overlap of cores (line 7). If the IoU is larger than the threshold, we remove the clip with the lower score from the list (lines 8–10). The removed clips will not contribute to further operation. Note that applying H-NMS with higher threshold could lead to a drop on accuracy due to aggressive suppression, while suppressing nearby clips with a lower threshold would increase the false alarm since the less confident proposals are less likely to be suppressed. To some extents, the threshold value makes a tradeoff between two conflicting needs. In our experiment, the IoU threshold value is set to 0.7.

Compared to conventional non-maximum suppression method, our proposed method takes advantage of the structural relation between the core region and clips, thus can avoid error dropout during the training. More importantly, the H-NMS does not harm the ultimate detection accuracy but substantially reduces the number of proposals. An example is shown in Fig. 4.5,

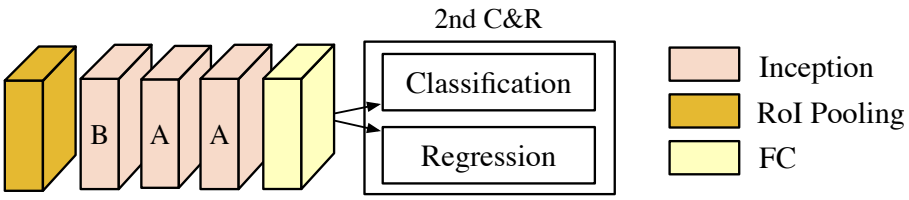


Figure 4.6 Tensor structure of Refinement.

the clip with 0.5 classification score (CS) is removed in conventional methods, while saved in our method if we consider the core within each clip.

4.3.3 Refinement

After the prediction of the first classification and regression in the clip proposal network stage, we get a rough prediction on hotspot localization and filtered region which is classified as non-hotspot. While the greedy method of clip filtering cannot guarantee all the reserved clips are classified correctly, the false alarm may be too high. To bring a robust detection with lower false alarms, we further construct refinement stage in the whole neural network, which includes a region of interests (RoI) pooling layer, three inception modules, as well as another classification and regression. The structure of Refinement is shown in Fig. 4.6.

RoI Pooling. The coordinates of each clip are the actual location from the original input image. We scale down the coordinates to conform with the spatial extent of the last feature map before the refinement. In traditional image processing, the most common ways to resize the image are cropping and warp-

ing. The crop method cuts the pattern boundary to fix the target size which leads to information loss. The warping operation will change the shape of origin features. Here we apply region of interests (RoI) pooling to transform the selected feature region $h \times w$ to a fixed spatial size of $H \times W$ (H and W are the hyper-parameters, and we use 7×7 in this work). For each pooled feature region $\lfloor h/H \times w/W \rfloor$, the max-pooling is applied independently. More specifically, the scaling is done by the following two steps. Firstly, each clip proposal is divided into equal-sized sections, the number of which is the same as the dimension of the output. Afterward, the largest value is found and output in each section. Consequently, the dimension of the output does not depend on the size of the input feature map nor on the size of the clips. On the contrary, it is determined solely by the number of sections we divide the proposal into (i.e. $H \times W$ in our algorithm). The RoI pooling transforms clips with different sizes into a fixed size which reserves the whole feature information and makes further hotspot classification and regression feasible. It bridges the two stages of our region-based hotspot detector, thus training object detection systems in an end-to-end manner is allowed. Additionally, it also benefits the processing speed in both the training and testing stage. Fig. 4.7 gives an example of RoI pooling operations.

Besides classification and regression in clip proposal network, here additional classification and regression are designed to fine-tune the clip location and give a more reliable classification re-

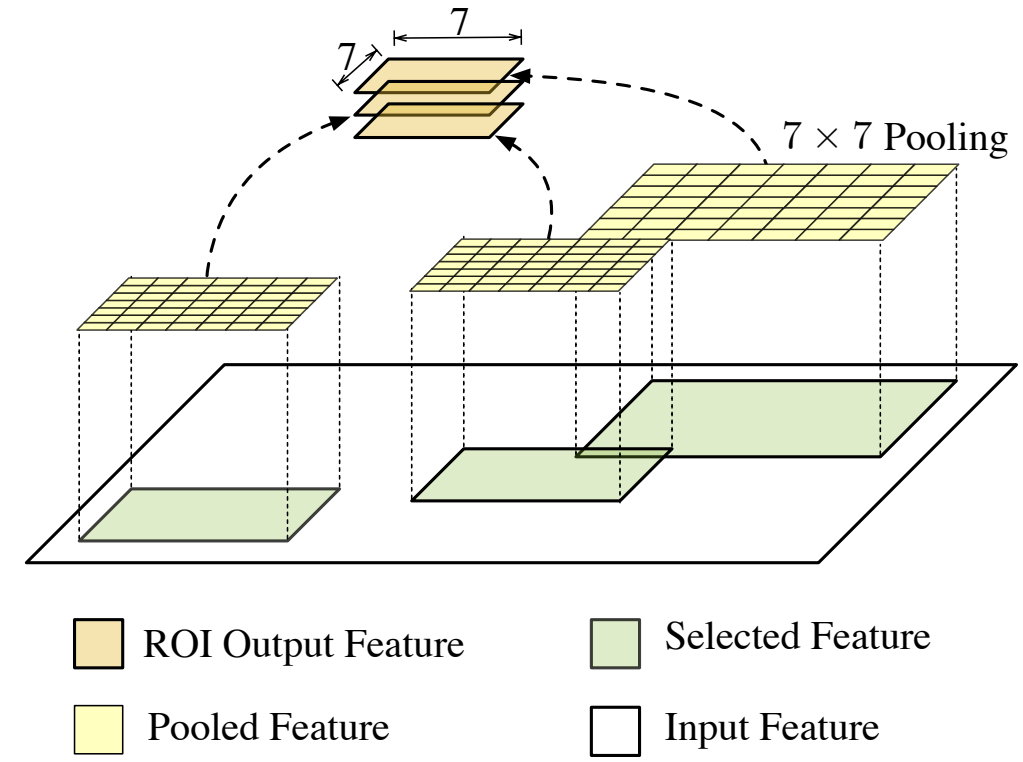


Figure 4.7 Visualized 7×7 RoI pooling.

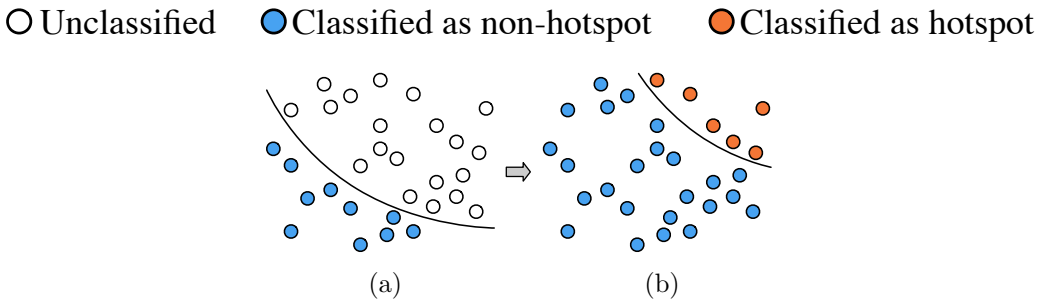


Figure 4.8 (a) 1st hotspot classification in clip proposal network; (b) The labelled hotspots are fed into 2nd hotspot classification in refinement stage to reduce false alarm.

sult. At this stage, most non-hotspot clips have been removed, thus two stage of hotspot classification can efficiently reduce false alarm. Fig. 4.8 illustrates an example of the two stage

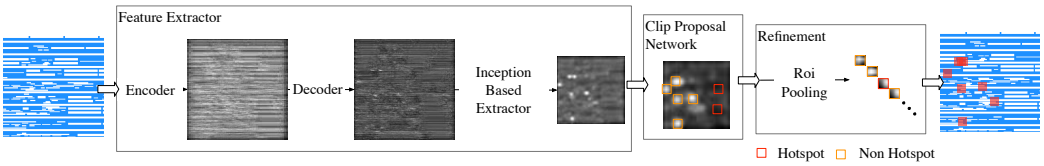


Figure 4.9 An example of features presentation in our framework.

hotspot classification.

4.3.4 Loss Function Design

We design a multi-task loss function called classification and regression (C&R) to calibrate our model. As shown in Fig. 4.4 and Fig. 4.6, C&R is applied both in the clip proposal network stage and refinement stage. The input tensors of 1st C&R are boxes in Fig. 4.4. W, H and C are width, height and channel respectively. The probability score of the hotspot, non-hotspot and prediction of clip coordinates are grouped in the channel direction. As aforementioned, x, y are the coordinates of the hotspot, which means the centre of the clip area. w, h are the width and height of the clip. In 2nd C&R, the tensor flow of the classification and regression is the same as [49] using fully-connected layers.

In the task of region-based hotspot detection, h_i is the predicted probability of clip i being a hotspot. h'_i is the ground truth of clip i, which equals to 1 if a hotspot is located in the centre and 0 vice versa. $\boldsymbol{l}_i = \{l_x, l_y, l_w, l_h\} \in \mathbb{R}^4$ and $\boldsymbol{l}'_i = \{l'_x, l'_y, l'_w, l'_h\} \in \mathbb{R}^4$ are assigned as coordinates of clips with in-

dex i representing the encoded coordinates of the prediction and ground truth respectively. The encoded coordinates can be expressed as:

$$\begin{aligned} l_x &= (x - x_g)/w_g, \quad l_y = (y - y_g)/h_g, \\ l'_x &= (x' - x_g)/w_g, \quad l'_y = (y' - y_g)/w_g, \\ l_w &= \log(w/w_g), \quad l_h = \log(h/h_g), \\ l'_w &= \log(w'/w_g), \quad l'_h = \log(h'/h_g), \end{aligned} \tag{4.5}$$

where variables x, x_g and x' are for the prediction of clip, g-clip and ground truth clip respectively (same as the y, w and h).

The classification and regression loss function for clips can be expressed as:

$$\begin{aligned} L_{C\&R}(h_i, \boldsymbol{l}_i) = \alpha_{loc} \sum_i h_i^{'} l_{loc}(\boldsymbol{l}_i, \boldsymbol{l}'_i) + \sum_i l_{hotspot}(h_i, h_i^{'}) \\ + \frac{1}{2}\beta(\|\boldsymbol{T}_{loc}\|_2^2 + \|\boldsymbol{T}_{hotspot}\|_2^2), \end{aligned} \tag{4.6}$$

where β is a hyper-parameter which controls the regularization strength. α_{loc} is the hyper-parameter which controls the balance between two tasks. The term $h_i^{'} l_{loc}(\boldsymbol{l}_i, \boldsymbol{l}'_i)$ indicates that regression loss is only activated for clips labeled as hotspots. $\boldsymbol{T}_{loc}$ and $\boldsymbol{T}_{hotspot}$ are the weights of the neural network. For elements $l_i[j]$ and $l'_i[j]$ $(j \in [1, 4])$ in $\boldsymbol{l}_i, \boldsymbol{l}'_i$ respectively, l_{loc} can be expressed as

$$l_{loc}(l_i[j], l'_i[j]) = \begin{cases} \frac{1}{2}(l_i[j] - l'_i[j])^2, & \text{if } |l_i[j] - l'_i[j]| < 1, \\ |l_i[j] - l'_i[j]| - 0.5, & \text{otherwise}, \end{cases} \tag{4.7}$$

which is the so-called robust loss or smooth L_1 loss (defined in [34]) applied to avoid the exploding gradients problem at training stage. $l_{hotspot}$ is the cross-entropy loss which is calculated as:

$$l_{hotspot}(h_i, h_i^{'}) = -(h_i \log h_i^{'} + h_i^{'} \log h_i). \tag{4.8}$$

In Equation (4.6), we apply the L_2 regularization to the loss function, which is the sum of the squares of all the weights in the network. The L_2 regularization penalizes peaky weight vectors and prefers diffuse weight vectors. Due to multiplicative interactions between weights and features, the L_2 regularization term has appealing property of encouraging the network to use all of its inputs rather than skewed on partial of its inputs.

4.3.5 Example of Detection Flow

An example of R-HSD flow is illustrated in Fig. 4.9. We first extract output tensors of each stage and sum up in channel-wise for visualization. Note that we visualize the features in grayscale, where the locations with lighter colors have higher values and darker locations have lower values vice versa. With feature extraction going deeper, values of features at hotspot regions have higher response comparing to the non-hotspot regions. The hotspot and non-hotspot areas presented as rectangles (for a clear explanation, not all rectangles are shown in the figures) in clip proposal network are cropped and downsampled to the same size with RoI pooling at the refinement stage. After

the second stage classification and regression, a more accurate result is given.

4.4 Enhanced R-HSD Neural Network

In previous sections, the preliminary R-HSD neural network has been proposed. However, there still exists some room to improve the performance of our region based hotspot detector. For example, the encoder-decoder structure in feature extractor becomes a bottleneck since it lacks a multi-level description of an input layout. Therefore, based on the preliminary design for R-HSD Neural Network, two new concepts are introduced for further enhancement.

4.4.1 Multi-branch Design for Encoder

The encoder-decoder in prior arts is in a single-branch structure, which may bring the following issues into the learning process. One is a fixed-size kernel cannot capture multi-scale information. The other is naively stacking large convolution operations is computationally expensive.

To alleviate the above issues, we propose our multi-branch design based on the basic idea of Inception network [83] and atrous convolution [7–9]. The core of the proposed design is that convolutional kernels with multiple sizes operate on the same level simultaneously. By aggregating the feature maps on different scales, the encoder-decoder structure has the potential

to surpass other feature extractors. Furthermore, to reduce the computational complexity, we exploit atrous convolution as an alternative to the standard convolution. As a generalization of standard convolution operation, atrous convolution is a powerful technique that explicitly controls the resolution of features computed by CNNs and adjust kernel's field of view to handle multi-scale information. Except for the dilated rate, it works in a similar way as standard convolution which moves across the whole image with a stride-size column change on the horizontal movements, and a stride-size row change on the vertical movements. We visualize the computation process of atrous convolution in Fig. 4.10, where only grids filled with dashed and oblique lines need to compute. It can be seen that the dilated rate controls the field-of-view scope of a kernel and affects the resolution of the output feature map. Note that when dilated rate equals to 1, the atrous convolution degrades to standard convolution. Hence, we can rewrite Equation (4.2) as following:

$$\begin{aligned} &\mathbf{F} \otimes \mathbf{K}(j,k) \\ &= \sum_{i=1}^{c} \sum_{m_0=1}^{m} \sum_{n_0=1}^{m} \mathbf{F}(i, j - r * m_0, k - r * n_0)\mathbf{K}(m_0, n_0), \end{aligned} \tag{4.9}$$

where r refers to the dilated rate. Implicitly expressed by Equation (4.9), the advantage of atrous convolution is that it expands kernel size without introducing additional computational complexity.

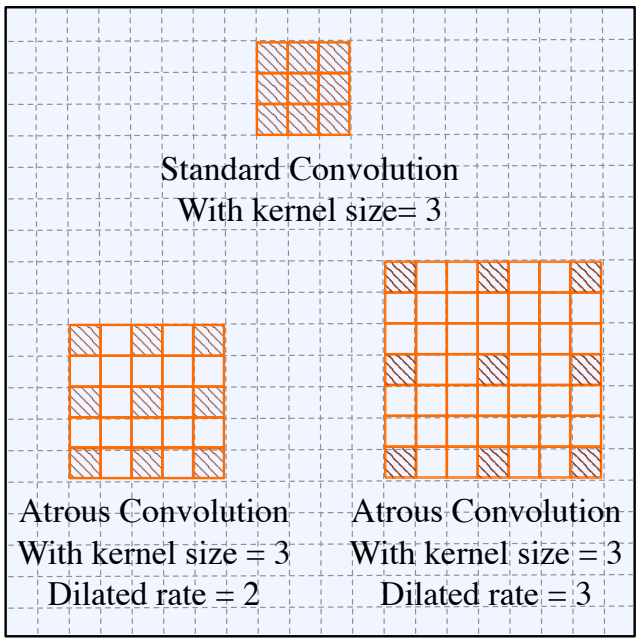


Figure 4.10 The illustration of atrous convolution.

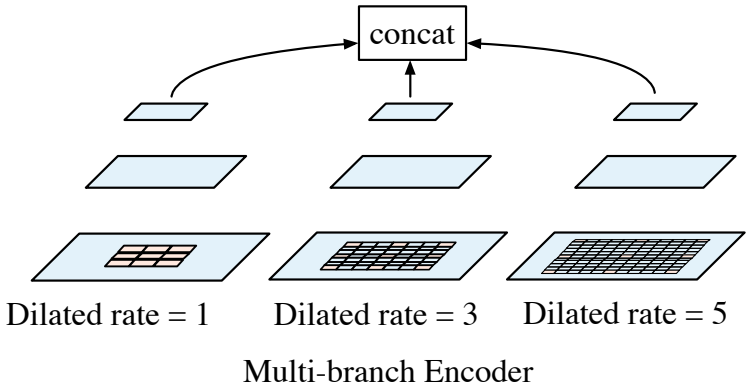


Figure 4.11 The Illustration of proposed multi-branch design.

Building on top of the aforementioned ideas, our multi-branch framework is designed as in Fig. 4.11. Three branches with different dilated rates (e.g. 1, 3, 5 as fine-tuned configurations) work collaboratively, and then all output tensors concatenate as a fusion feature map via channel dimension.

4.4.2 IoU Regularizer for Loss Function Design

To accelerate bounding box prediction, we leverage a novel IoU [74,112] regularization term in our loss function. Hence, our loss function is redesigned as:

$$\begin{aligned} L_{C\&R}(h_i, \boldsymbol{l}_i) = \alpha_{loc} \sum_i h_i^{'} l_{loc}(\boldsymbol{l}_i, \boldsymbol{l}_i^{\prime}) + \sum_i l_{hotspot}(h_i, h_i^{'}) \\ + \frac{1}{2}\beta(\|\boldsymbol{T}_{loc}\|_2^2 + \|\boldsymbol{T}_{hotspot}\|_2^2 + R_{IoU}), \end{aligned} \tag{4.10}$$

where R_{IoU} is the proposed IoU regularizer.

IoU, also known as the Jaccard index, is a widely exploited metric for comparing the similarity between two geometric shapes. IoU encodes the shape and position properties of the objects under comparison, e.g. the indices of left upper corner and right bottom corner of two clips in our case, into the region property, and then calculates a normalized measure that focuses on their areas. This property makes IoU robust to the scale of the problem under consideration. Thanks to this appealing property, this metric is the foundation of all performance measures in segmentation, object detection, and tracking tasks. The exploited IoU regularizer, shown in Fig. 4.12, directly enforces the maximal overlap between the predicted clip and the ground truth, and jointly regresses all the bound variables as a whole unit.

To give a more mathematical understanding of the proposed IoU regularizer, the deduction of back-propagated information of itself is shown as follows. Assume the predicted clip and corresponding ground truth are located as shown in Fig. 4.12.

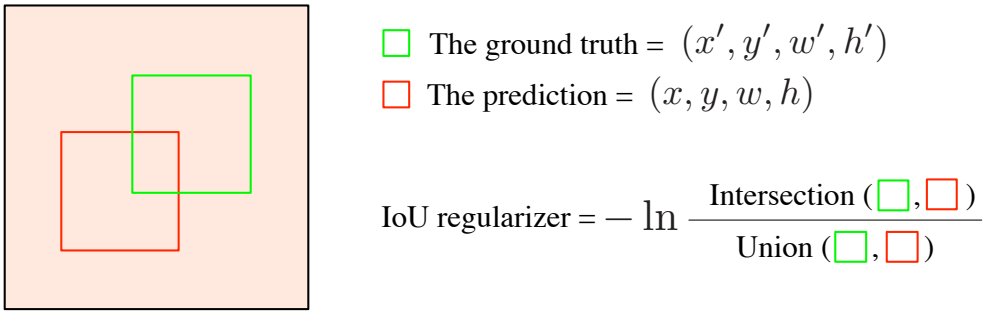


Figure 4.12 The illustration of the IoU regularizer.

Table 4.1 Benchmark information. `case2`, `case3` and `case4` are three cases from ICCAD CAD contest 2016 benchmark suite [90]. `Via` is generated by open source layout generator and simulated using Mentor Calibre. Clips for training are generated by random cropping on layouts.

Bench	Train #HS	Test #HS	Train #Clips†	Test #Clips	Training Set Size ($\mu m \times \mu m$)	Testing Set Size ($\mu m \times \mu m$)
case2	40	39	1000	8	6.95 × 3.75	6.95 × 3.75
case3	1388	1433	1000	33	12.91 × 10.07	12.91 × 10.07
case4	90	72	1000	55	79.95 × 42.13	79.95 × 42.13
Via	2184	2184	1000	1947	53.82 × 53.82	53.82 × 53.82

Firstly, the partial derivatives of the area of the predicated clip with respect to y_l, y_r, x_l, x_r, are computed as:

$$\frac{\partial X}{\partial y_l \,(\textbf{or}\ \partial y_r)} = x_r - x_l, \tag{4.11}$$

and

$$\frac{\partial X}{\partial x_l \,(\textbf{or}\ \partial x_r)} = y_r - y_l. \tag{4.12}$$

For simplicity, we use ∇X refer to any derivatives w.r.t y_l, y_r, x_l, x_r. Next, the partial derivatives of the intersection area I w.r.t any y_l, y_r, x_l, x_r, marked as ∇I, is deduced as:

$$\frac{\partial I}{\partial y_l} = x_r - x_l', \tag{4.13}$$

while $\frac{\partial I}{\partial y_r} = 0$ and

$$\frac{\partial I}{\partial x_r} = y_r' - y_l, \tag{4.14}$$

while $\frac{\partial I}{\partial x_l} = 0$. Eventually, the back-propagated information of proposed regularization w.r.t p is

$$\begin{aligned} \nabla R_{IoU} &= \frac{I\left(\nabla X - \nabla I\right) - U\nabla I}{U^2 IoU} \\ &= \frac{1}{U}\nabla X - \frac{U+I}{UI}\nabla I, \end{aligned} \tag{4.15}$$

where the union area U equals to $(X + X')$. According to Equation (4.15), the first term $\frac{1}{U}\nabla X$ penalizes the predicted clip, whilst the second term is a soft constraint on the intersection area. When the gradient equals to zero, the limit case which means predicted clip exactly matches the ground truth are attained.

4.5 Experimental Results

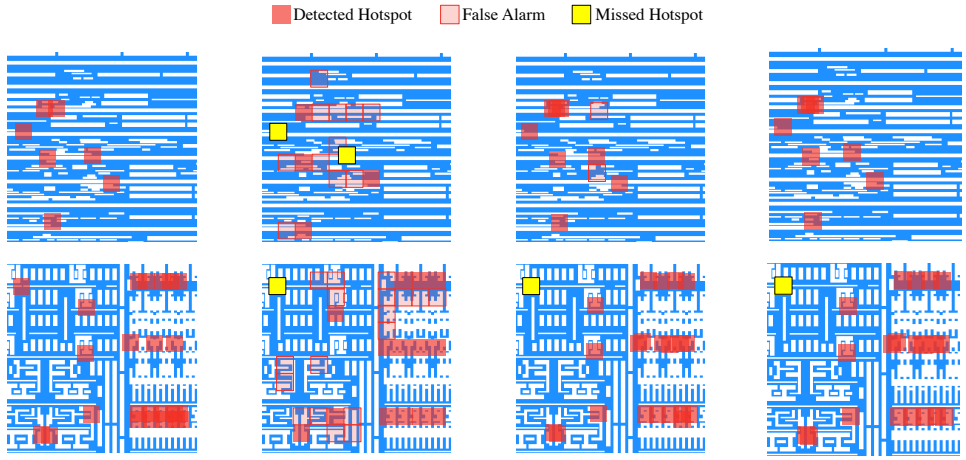
Our region-based hotspot detection flow is evaluated on ICCAD CAD Contest 2016 benchmark suite [90], which contains four designs that are shrunk to match EUV metal layer design rules. Ground truth hotspot locations are labelled according to the results of industrial $7nm$ metal layer EUV lithography simulation under a given process window[1]. As there are limited defects found with lithography simulation on the first benchmark, all

[1]Shrunk benchmarks and their hotspot information is available at https://github.com/phdyang007/ICCAD16-N7M2EUV

our experiments are conducted on rest three designs. Each layout is split into two equal halves with one part used for training and the other one used for testing. Besides these three cases, we generate a much larger benchmark called `Via` to present a more comprehensive comparison with previous related works. `Via` benchmark is generated following an open source layout generator[2], and simulated using Mentor Calibre. More details about benchmarks are shown in TABLE 4.1. We implement our region based hotspot detection framework with Tensorflow [1] in Python, and test it on a platform with a Xeon Silver 4114 processor and a Nvidia GTX Titan graphic card. Nvidia GTX Titan has a comparable computational capacity as the high performance clusters with 24 Maxwell stream processors. *Note that three training layouts are merged together to train one model that will be used in the inference stage.* In the following experiments, the neural network is trained with following parameter settings: $input\ size = 256 \times 256$ (corresponding to $2.56\mu m \times 2.56\mu m$), $batch\ size = 12$, $initial\ learning\ rate = 0.002$ (decay ten times for each 30000 steps), $aspect\ ratio = [0.5, 1.0, 2.0]$ and $scales = [0.25, 0.5, 1.0, 2.0]$. The parameters of the loss function are heuristically chosen, what we use are $\beta = 0.2, \alpha_{loc} = 2.0$. At the inference stage, we follow the same data generation rule applied in [102].

We list the detailed result comparison in TABLE 4.2. Column "Bench" lists three benchmarks used in our experiments.

[2] https://github.com/phdyang007/layout-generator



(a) Ground-truth (b) TCAD'19 [102] (c) R-HSD [12] (d) Enhanced R-HSD

Figure 4.13 Visualization of different hotspot detection results.

Columns "Accu", "FA", "Time" denote hotspot detection accuracy, false alarm count and detection runtime respectively. Column "TCAD'19" lists the result of a deep learning-based hotspot detector proposed in [102] that adopts frequency domain feature extraction and biased learning strategy. We also implement two baseline frameworks that employ Faster R-CNN [71] and SSD [57], respectively, which are two classic techniques match our region-based hotspot detection objectives well. Note that we do not apply the pre-trained model in this work, all the models are trained from scratch. The corresponding results are listed in columns "Faster R-CNN [71]" and "SSD [57]". The rest two columns, "R-HSD" and "Enhanced R-HSD", denote the methods proposed in [12] and the framework presented in this work. The results in R-HSD surpass [102] with average of 6.11% improvement on hotspot detection accuracy and $\sim$ 170 less false

alarm penalty, while our enhanced R-HSD behaves even better with an average accuracy of 94.97 % and further decrease on the false alarm compared to the R-HSD. Especially, our framework gets much better on `case2` with 95.74% detection accuracy compared to 77.78%, 1.8% and 71.9% for [102], Faster R-CNN, and SSD respectively.

The advantage of the proposed two-stage classification and regression flow can also be seen here that [102] achieves similar hotspot detection accuracy compared to our frameworks but has extremely large false alarms that will introduce additional issue. As shown in TABLE 4.3, the detection runtime for the proposed region-based framework is much faster than [102] thanks to the region-based detection scheme. We can also observe that although Faster R-CNN and SSD are originally designed for large region object detection, they perform very poor on hotspot detection tasks which reflects the effectiveness and efficiency of our customized frameworks. Different from the atrous spatial pyramid pooling (ASPP) proposed in [7,9] which extracts features in multiple scales. Multi-branch encoder with atrous convolution layer designed for clip-wise feature extractions has much lower dilation rate than ASPP, because the size and scale of clips are much regular than objects in real life. The setting of ASPP is not compatible with the hotspot detection task. The experiments in TABLE 4.4 show that our proposed design in hotspot detection task outperforms ASPP by a large margin.

We also study how different configurations of our framework

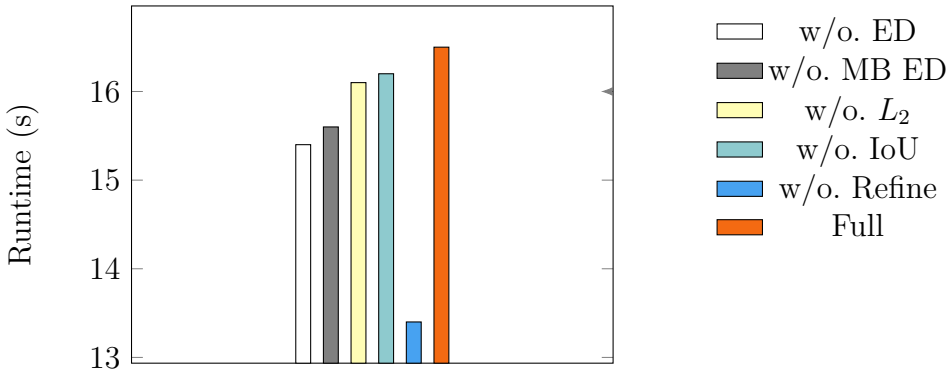


Figure 4.14 Runtime comparison among different settings.

affect performance. TABLE 4.5 and Fig. 4.14 summarize the contributions of encoder-decoder structure, multi-branch encoder-decoder, L_2 regularization, IoU regularizer and refinement stage to our backbone neural network. "w/o. ED" denotes the framework without the encoder-decoder structure, "w/o. MBED" denotes the framework without the multi-branch design for the encoder-decoder structure, "w/o. L_2" stands for the framework without the L_2 regularization, "w/o. IoU" denotes the framework without the IoU regularizer, "w/o. Refine" denotes the framework without the refinement classification and regression, and "Full" is our framework with entire techniques. The ablation study shows that with the encoder-decoder structure, we get 7.17% accuracy improvement on average, which indicates that the encoder-decoder structure gives a more efficient feature expression than the original input. After incorporating multi-branch design for the encoder-decoder structure, the accuracy is improved by 1.82% on average, which demonstrates the effectiveness of this configuration. It can be seen that without IoU

regularizer, the performance degrades with 45 additional false alarms. With the L_2 regularization, the framework gets around 3% improvement in all cases, which means under the same experiment settings, the L_2 regularization resolves the overfitting problem effectively. Comparing the whole framework with the model without refinement, the model with refinement reduces around 20% false alarms and achieves 5.25% further improvement on average accuracy. TABLE 4.6 shows an ablative comparison on different anchor generation settings. "Anchor number" denotes the number of anchors we generate for each location. "Scales" denotes the size ratio compared to a standard anchor with the shape of 16×16. "Aspect Ratios" denotes the ratio between anchor width and height. With the increasing of anchors, the number of false alarms can be reduced signficantly, which indicates a sufficient sampling is necessary for the training.

4.6 conclusion

In this paper, we have proposed an innovative end-to-end region-based hotspot detection framework. Our feature extractor based on multi-branch encoder-decoder design and inception module provides a self-adaptive way to perform feature transformation, which is very compatible with convolution neural networks. With pruning and hotspot non-maximum suppression strategies, the clip proposal network locates the potential hotspot in an effi-

cient regression way. We take advantage of L_2 regularization's property to prevent over-fitting and get higher performance. IoU regularizer is leveraged to boost the regression procedure thus attain improvements on both accuracy and false alarm. Additionally, our classification and regression strategy with refinement reduces false alarms and increases accuracy at a remarkable speed. The experimental results show that our framework outperforms the current deep learning based models. The defect results in this paper come from rigorious EUV model simulation model. In our future work, more experiments with compact lithography simulation will be discussed. We hope this paper can give a new perspective on deep learning based hotspot detection and provide a more powerful solution for the advanced design for manufacturability (DFM) research.

□ **End of chapter.**

Table 4.2 Performance comparison with state-of-the-art methods. Faster R-CNN [71] and SSD [57] are two classical techniques match to the region-based hotspot detection objectives. TCAD'19 [102] is a deep learning based hotspot detector. R-HSD and Enhanced R-HSD are the methods proposed in [12] and in this work, separately.

Bench	TCAD'19 [102]			Faster R-CNN [71]			SSD [57]			R-HSD [12]			Enhanced R-HSD		
	Accu (%)	FA	F1	Accu (%)	FA	F1	Accu (%)	FA	F1	Accu (%)	FA	F1	Accu (%)	FA	F1
case2	77.78	48	0.51	1.80	3	0.05	71.90	519	0.53	93.02	17	0.78	95.74	15	0.81
case3	91.20	263	0.87	57.10	74	0.11	57.40	1730	0.61	94.50	34	0.96	94.72	78	0.94
case4	100	511	0.22	6.90	69	0.07	77.80	275	0.03	100	201	0.38	100	92	0.61
Via	81.30	2672	0.54	86.52	25551	0.12	64.21	65319	0.38	87.20	2577	0.57	89.40	2435	0.58
Average	87.57	873.5	0.54	38.08	6424.5	0.24	67.83	16960	0.39	93.68	707.3	0.67	94.97	655	0.74
Ratio	1.00	1.00	1.00	0.43	7.35	0.44	0.77	19.42	0.72	1.07	0.96	1.24	**1.08**	**0.91**	**1.37**

Table 4.3 Runtime Comparison with State-of-the-art methods.

Bench	TCAD'19 [102]	Faster R-CNN [71]	SSD [57]	R-HSD [12]	Enhanced R-HSD
case2	60.0	1.0	1.0	2.0	2.3
case3	265.0	11.0	3.0	10.0	10.8
case4	428.0	8.0	2.0	6.0	6.6
Via	87.8	57.1	21.3	43.5	46.3
Average	210.2	19.3	6.8	15.4	16.5

Table 4.4 Comparison with ASPP Module. ASPP module is approached in [9], which is designed for general objects.

Bench	ASPP [9] + R-HSD [12]				Enhanced R-HSD			
	Accu (%)	FA	F1	Time (s)	Accu (%)	FA	F1	Time (s)
case2	82.05	22	0.68	3.6	95.74	15	0.81	2.3
case3	90.95	96	0.92	11.0	94.72	78	0.94	10.8
case4	100	143	0.50	4.45	100	92	0.61	6.6
Via	87.61	2558	0.58	44.7	89.4	2435	0.58	46.3
Average	90.39	704.8	0.67	15.38	94.97	655	0.74	16.5
Ratio	1.00	1.00	1.00	1.00	**1.05**	**0.97**	**1.10**	1.07

Table 4.5 Ablation study on each proposed strategy.

Bench	w/o. ED		w/o. MB ED		w/o. L_2		w/o. IoU		w/o. Refine		Full	
	Accu (%)	FA	Accu (%)	FA	Accu (%)	FA	Accu (%)	FA	Accu (%)	FA	Accu (%)	FA
case2	75	31	93.32	20	91.9	1	94.11	21	75.76	394	95.74	15
case3	92.8	27	94.72	31	91.7	64	94.91	23	94	23	94.72	78
case4	98.6	166	100	220	97.2	127	100	240	100	201	100	92
Via	85.12	2762	84.57	2812	86.5	2697	88.7	2516	89.1	2516	89.4	2435
Average	87.8	746.5	93.15	770.8	91.82	722.3	94.43	700	89.72	783.5	94.97	655
Ratio	0.925	1.06	0.981	1.07	0.967	1.04	0.994	1.07	0.94	1.20	**1.00**	**1.00**

Table 4.6 Ablation study on anchor generation.

Anchor number	Scales	Aspect Ratios	Average Accu (%)	Average FA
3	[0.25, 0.5, 1.0]	[0.5]	89.13	3244.5
6	[0.25, 0.5, 1.0]	[0.5, 1.0]	92.31	1820
9	[0.25, 0.5, 1.0]	[0.5, 1.0, 2.0]	90.12	1725.5
12 (final result)	[0.25, 0.5, 1.0, 2.0]	[0.5, 1.0, 2.0]	94.97	1655

Chapter 5

Layout Pattern Analysis

5.1 Introduction

The yield of manufactured integrated circuits (ICs) is defined as the percentage of good dies among all dies manufactured. A high and stable yield could ensure profitability and reliability of products. However, as the feature size decreases, specific layout patterns that are hard to fabricate tend to cause more systematic defects, such as open or bridge defects in neighboring wires. These layout patterns are an important source of yield loss. Since layout configurations of new designs may differ from existing ones, identifying layout patterns that lead to yield loss through test chips, SRAMs, etc., is becoming less effective. Performing hotspot detection [11, 102, 108] on entire layouts may result in overcorrection which can adversely affect chip area and performance. Physical failure analysis (PFA) is a straightforward method to determine whether a layout pattern is the root cause of systematic defects. However, it requires both experi-

ence and a proper understanding of the fabrication process and is usually time-consuming and expensive.

To efficiently identify the root cause of systematic defects, statistical methods have been adopted to automatically identify common physical defect features by analyzing volume diagnosis reports. One of the most prominent work is a Bayesian method proposed in [3], which characterizes the conditional distribution of systematic defect given potential root causes. It learns the optimal root cause distribution by maximizing the likelihood of observed diagnosis report using an Expectation-Maximization (EM) learning algorithm. There are also works [18,87] focusing on improving the quality of diagnosis results by evaluating the impact of diagnosis features to improve the root cause identification accuracy. These methods fall short of considering root cause layout patterns which largely restricts their applicability to real tasks. Cheng *et al.* [17] proposed an advanced solution based on [3]. They take root cause layout patterns into consideration when identifying the correct layout patterns inducing systematic yield loss. In practice, there usually exists complex interactions between different root causes, as well as root cause and systematic defect, but the causal relationship between candidate layout patterns and the systematic defect was not considered in [17].

Another class of seminal works [85,86] focus more on the geometric structure of layout patterns by adopting clustering algorithms to improve the systematic IC-defect identification. Both

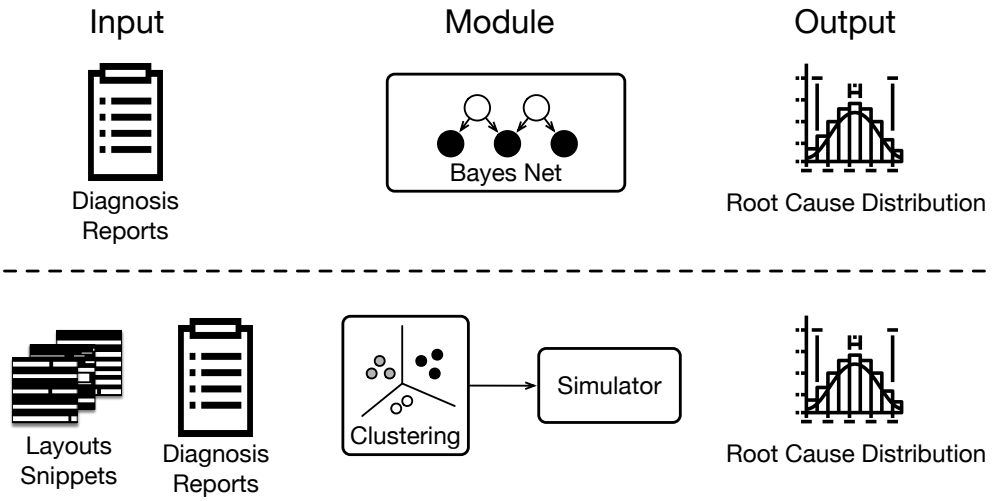


Figure 5.1 Overview of prior methods. Upper: [3, 18]; lower: [85, 86].

connectivity-based and centroid-based clustering algorithms are used to group rotation-, mirror-equivalent layout snippets together. These works conduct clustering on raw layout snippets in a two-stage manner and manually check all possible geometric equivalence between different clusters. Simulation experiments in [85] indicate that the resolution of the identification results of clustering-based methods is limited, since they may require a failure analysis expert's judgment to pick a single layout snippet for each cluster. Furthermore, layout snippets that are shift-equivalent are regarded as different candidates, which is commonly considered unreasonable since these snippets share identical or similar geometric structures.

To address the above issues and improve the resolution of root cause identification, a unified framework for layout pattern analysis with deep causal effect estimation is proposed in this

work. Compared to existing statistical learning methods, our framework characterizes the causal relationship between potential root causes and the systematic defect. Compared to methods using clustering algorithms, our framework regards rotation-, mirror-, and shift-invariant layout snippets as equivalent without requiring any manual equivalence check. At the first stage of the framework, a novel contrastive learning method is used to train an encoder network to extract from layout snippets their rotation-, mirror-, and shift-invariant latent features. The latent features are then clustered to form layout patterns. Based on the learned layout patterns, we use a Structural Causal Model (SCM) to model the causal relationship between candidate layout patterns and the systematic defect, i.e. the model describes the relationship between the occurrence/presence of a certain layout pattern and the systematic defect. Lastly, the Average Causal Effect (ACE) of candidate layout patterns on the systematic defect is estimated as the metric for the identification of the root cause of systematic defects. Experimental results on large-scale designs show that our framework achieves state-of-the-art results which significantly outperforms a commercial tool in terms of accuracy as well as inference time.

To the best of our knowledge, this work is the first to apply contrastive learning-based deep learning techniques and average causal effect estimation to identify the root cause. The main contributions of this work are threefold:

- 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. A novel clustering loss is proposed to solve the limitation of conventional contrastive learning method. Our framework can identify the critical root causes and provide high-resolution clustered snippets for further analysis.
- The causal relationship between different candidate layout patterns and the systematic defects is characterized using a neural network and a neural network attribution method is adopted to estimate the average causal effect for root cause identification.
- Experimental results on several industrial designs show the effectiveness and robustness against the noise of our framework. The accuracy of our framework outperforms a commercial tool and state-of-the-art framework in different scenarios and we get $\times$8.4 speedup on average at inference.

The remainder of this paper is organized as follows. Section 6.2 introduces terminologies and problem formulation related to this work. Section 6.3 describes the problem formulation and the algorithmic details of our framework. Section 6.4 lists the experimental results, followed by the discussion and conclusion in Section 6.5.

5.2 Preliminaries

In this section, preliminary knowledge related to the proposed framework is briefly reviewed.

5.2.1 Layout Pattern Analysis

Layout Pattern Analysis (LPA) takes a step towards identifying the layout patterns which cause systematic defects. Cheng *et al.* [17] proposed an LPA solution which is an enhanced flow based on [3]. Attribute to the external steps on layout pattern processing, this work makes root cause identification on layout patterns becomes feasible. The challenge of how to handle a large number of potential layout patterns to be considered for analysis is solved. And the risk of over-fitting caused by Bayesian modeling is also addressed. Layout pattern extraction is designed to extract all unique layout patterns around locations that could be physical defects. In layout pattern matching, layout patterns are transformed to canonical forms which make shifted, rotated, or mirrored patterns identical. Combining previous steps with the root cause identification method proposed in [3], root cause analysis results including root cause distribution and layout patterns are returned for further study.

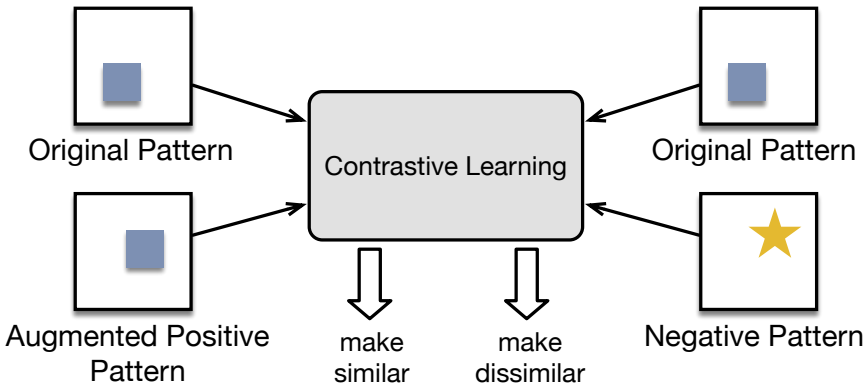


Figure 5.2 Idea of contrastive learning: maximize the similarity between latent features of an image and its augmented version and simultaneously minimize the similarity between latent features of inputs correspond to different original images.

5.2.2 Contrastive Learning

Conventional deep networks training often relies on large amounts of annotated data to learn representations in a latent space. Since the annotated data can be costly or even impossible to collect, self-supervised learning leverages unlabeled data to perform pretext tasks for representation learning [22, 62]. Contrastive learning is a class of self-supervised learning that uses contrastive objectives. The general idea of contrastive learning is to maximize the similarity between an instance and its augmentation, while keep the discriminative power against different instances through a contrastive loss in the latent space, as illustrated in Fig. 5.2. Recent contrastive learning methods [13,14,26,36] have achieved competitive results in visual representation learning compared with prominent supervised learn-

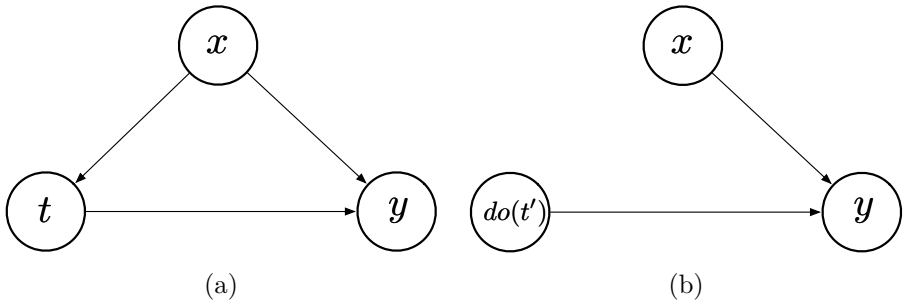


Figure 5.3 An SCM (a) without and (b) under intervention. Nodes represent random variables and directed edges $x \rightarrow y$ indicates that x is a direct cause of y. Under intervention $do(t')$, the intervened variable t is fixed to the intervened value t' and all its incoming edges are removed.

ing methods for computer vision tasks.

5.2.3 Structural Causal Models

Structural Causal Models (SCMs) [67] are developed towards a comprehensive theory of causation and serve as a key ingredient of our framework.

Definition 4 (Structural Causal Model [67]). *A structural causal model M is a 4-tuple $(E, X, F, P(E))$, where*

- *E is a set of exogenous (unobserved) variables;*
- *X is a set of endogenous (observed) variables;*
- *F represents a collection of functions $F = \{f_i\}$ such that each endogenous variable $x_i \in X$ is determined by a function $f_i \in F$, where f_i is a mapping from the respective domain of $\epsilon_i \cup Pa_i$ to x_i, with $\epsilon_i \subseteq E$, $Pa_i \subseteq X \backslash \{X_i\}$ is the set of direct parents of x_i;*

- *The uncertainty is encoded through a probability distribution over the exogenous variables,* $P(E)$.

SCMs provide a compact way of characterizing average causal effect $ACE^{y}_{do(x_i)}$, which is defined as $\mathbb{E}[y|do(x_i = 1)] - \mathbb{E}[y|do(x_i = 0)]$ for binary x_i. $\mathbb{E}[y|do(x_i = \alpha)]$, known as *interventional expectation* [67], denotes the expectation of y when intervening the value of x_i to be α. For an SCM, such intervened model can be represented by replacing the structural equation $x_i = f_i(Pa_i, \epsilon_i)$ by a constant $x_i = \alpha$.

5.2.4 Neural Network Attributions

Attribution methods aim to provide interpretability of deep networks by identifying the effect of an input neuron on a specific output neuron [78, 81]. Recently, [5] approached neural network attribution problems from a causal perspective. They view a multilayer perceptron (MLP) $\{l_1, \ldots, l_n\}$ as an SCM $M'(E, \{l_1, l_n\}, f', P(E))$, where l_1 is the input layer, l_n is the output layer, E refers to a set of exogenous random variables which act as causal factors for the input neurons l_1, f' refers to the mapping from the input to output by marginalizing out all hidden neurons.

Based on SCM reformulation, [5] approximated the interventional expectation of the output neurons $f'(l_1)$ under the inter-

vention $do(x_i = \alpha)$ as:

$$\begin{aligned}&\mathbb{E}[f'(l_1)|do(x_i = \alpha)] \approx f'(\mu)+\\&\frac{1}{2}\text{tr}(\nabla^2 f'(\mu)\mathbb{E}[(l_1 - \mu)(l_1 - \mu)^T|do(x_i = \alpha)]),\end{aligned} \tag{5.1}$$

where $\text{tr}(\cdot)$ is the trace operator, $\mu = [\mu_1, \ldots, \mu_k]^T$ and each entry $\mu_{i'} = \mathbb{E}[x_{i'}|do(x_i = \alpha)]$, $\forall x_{i'} \in l_1$, is the interventional expectation of $x_{i'}$ when x_i is intervened to the value α.

5.3 Methodologies

5.3.1 Overview

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. 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.). Our framework identifies the true root cause(s) inducing systematic defects in R by exploiting both the geometric structure of layout snippets (Section 5.3.2) and causal relationship between potential root causes and systematic defect (Section 5.3.3).

An illustration of our LPA framework is given in Fig. 5.4. It uses diagnosis reports and layout snippets of potential root causes in these reports as the inputs. First, a contrastive learning-based method is adopted to extract rotation-, mirror-, and shift-

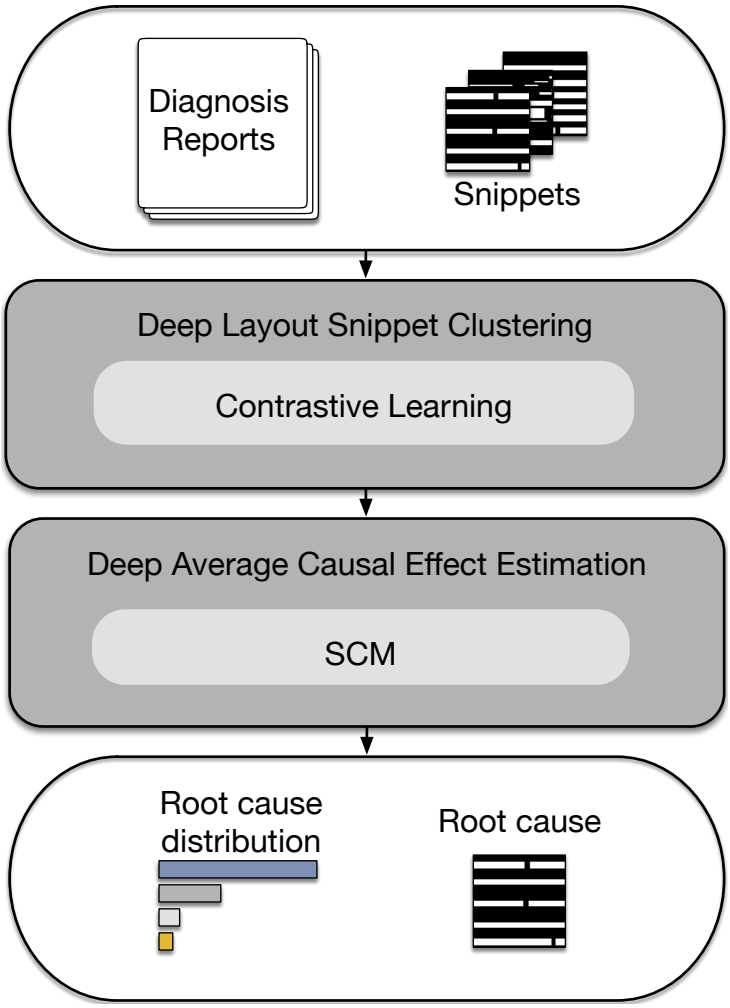


Figure 5.4 Overview of the framework.

invariant latent features from input layout snippets. Then, the latent features are clustered using k-means clustering to identify layout patterns from a large amount of layout snippets. Each cluster corresponds to one layout pattern. Third, a feed-forward neural network, which acts as the SCM involves all candidate layout patterns and systematic defect, is trained to maximize the likelihood of the input diagnosis reports given the representation of candidate layout patterns as inputs. After training, the Average Causal Effects (ACEs) of layout patterns to systematic defect are evaluated to identify the true root cause(s).

5.3.2 Deep Layout Snippet Clustering (DLSC)

The identification of layout patterns from layout snippets using clustering algorithm is elaborated in this section.

Since there are a considerable number of duplicate and equivalent layout snippets in the diagnosis reports, layout pattern matching is usually conducted to recognize layout snippets that are rotated, mirrored, or shifted version of each other as geometrically equivalent. Clustering algorithms are a widely-used class of techniques in layout pattern matching. Although applying connectivity-based or centroid-based clustering algorithms on raw layout snippets achieved certain improvements on identifying root causes, they heavily rely on manual design and may have difficulty when generalized to new manufacturing processes.

To circumvent the need of manually designed clustering rules, we introduce deep neural networks in layout pattern matching. Specifically, an encoder network is trained using contrastive learning to extract latent features that are invariant to trivial transformations such as rotation, mirror and shift. Besides, the self-supervised nature of contrastive learning allows us to construct a huge amount of training data set by cropping unlabeled layout snippets from the entire layout designs.

Encoder Network. The principle of the encoder network is to transform raw layout snippets into a low-dimensional latent space, in which equivalent layout snippets are mapped to an identical embedding (vector). The low dimensional embeddings represent prototypes of layout snippets. The network structure of our model is shown in Fig. 5.5. "SeparableConv" indicates depthwise separable convolution layer which is a variant convolution layer widely used in [39] for computation efficiency. "Block A" and "Block B" are two modules with residual connections. Three "Linear" layers are attached to the feature extractor as a bottleneck structure maps two-dimensional features to embeddings.

Contrastive Learning. Given a batch of embeddings transformed from raw layout snippets by the encoder network, our goal is to make the embeddings of equivalent layout snippets identical, while keeping those of non-equivalent as dissimilar as possible. To achieve this, ℓ_p-norm is used as a metric to measure the dissimilarity between embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ of a pair of

layout snippets:

$$d(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_p , \tag{5.2}$$

where p is a real number greater than 1 and is set to $p = 2$ in this work. Conventional contrastive loss [10, 93] based on the metric above is given by

$$L_{con}(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{n}) = \max(d(\boldsymbol{z}, \boldsymbol{p}) - d(\boldsymbol{z}, \boldsymbol{n}) + marg, 0), \tag{5.3}$$

where $marg$ is a non-negative value indicating an appropriately set margin, $\boldsymbol{z}$, $\boldsymbol{p}$, and $\boldsymbol{n}$ are the embeddings of one layout snippet, the embedding of a positive sample of $\boldsymbol{z}$, and the embedding of a negative sample of $\boldsymbol{z}$, respectively. $marg$ represents the minimum difference between positive and negative distances that is required for the loss to be zero. During training, positive samples $\boldsymbol{p}$ are getting closer to the anchor embedding $\boldsymbol{z}$ and negative samples $\boldsymbol{n}$ are penalized to be far from anchor embedding. An illustration on contrastive learning with an encoder is shown in Fig. 5.6.

Clustering Loss. According to Equation (5.3), the embedding clustering is improved by pair-wise comparison directly. While this method has drawbacks, the property of clusters is overlooked. An example of a small batch size with two clusters is presented in Fig. 5.7, the pair-wise operation may pull the positive samples away from the center of clusters. The double-headed arrow with a dotted line indicates the penalization term in Equation (5.3), this term causes two samples away from the center of corresponding clusters. This may cause negative ef-

fects on the efficiency of convergence and clustering quality. It will be more difficult for training when the batch size is larger, since more clusters might be disrupted. To tackle this issue, we propose a clustering loss by adding a regularization term to improve the quality of clusters. This term is expressed as:

$$R(\boldsymbol{m}_z, \boldsymbol{z}) = \|\boldsymbol{m}_z - \boldsymbol{z}\|_p , \tag{5.4}$$

where m_z is the cluster center of sample z. The whole optimization objective is expressed as:

$$L(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{n}, \boldsymbol{m}_p, \boldsymbol{m}_n) = L_{con}(\boldsymbol{z}, \boldsymbol{p}, \boldsymbol{n}) + \alpha R(\boldsymbol{m}_p, \boldsymbol{p}) + \beta R(\boldsymbol{m}_n, \boldsymbol{n}), \tag{5.5}$$

where α and β are the weights of regularization terms. With the regularization terms for positive samples and negative samples, the drawback of contrastive loss in [10] is avoided and the risk of low resolution is reduced. Each iteration of the training decreases the quantization error of the clusters since for the same cluster, we have the following inequality:

$$\sum_i R(\boldsymbol{m}', \boldsymbol{z}_i) - R(\boldsymbol{m}, \boldsymbol{z}_i) = \sum_i \|\boldsymbol{m}' - \boldsymbol{z}_i\|_p - \|\boldsymbol{m} - \boldsymbol{z}_i\|_p \leq 0,$$

where $\boldsymbol{m}'$ is the updated cluster center and $\boldsymbol{s}_i$ is the sample i belongs to the cluster. Repeatedly replacing $\boldsymbol{m}$ by $\boldsymbol{m}'$ speeds up the convergence of the training schedule. Experimental results verified that the proposed clustering loss requires less training data and computation resources. The advantage of clustering loss is summarized in TABLE 5.1. A visualized illustration of

Table 5.1 Advantage of Clustering Loss compared to Contrastive Loss used in [10].

	Cluster Awareness	Training Effciency	Low Data Requirement
Ours	✓	✓	✓
[10]	✗	✗	✗

Equation (5.5) is presented in Fig. 5.8. The detail of how to construct the positive and negative samples and training scheme is clarified in Section 6.4.

After training the encoder network, it is used to extract embeddings of layout snippets. Then k-means clustering algorithm is applied to these embeddings to partition them into k clusters C_i, $i \in \{1, \ldots, k\}$. Each cluster C_i consists of n_i equivalent layout snippets that correspond to one layout pattern. The silhouette method [76] which is a measurement of how similar an object is to its own cluster compared to other clusters is adopted to determine the optimal value of the number of clusters k. By embedding clustering, equivalent layout snippets can be grouped into the same cluster without artificial modulation.

An example of the DLSC is illustrated in Fig. 5.9, layout snippets with a large number of pixels are transformed to low-dimensional embeddings which reduces the clustering computation remarkably while improves the layout pattern matching accuracy. Experimental results in section 6.4 empirically show that encoder network trained using layout snippets of one layout design can also generalize to new layout designs.

5.3.3 Deep Average Causal Effect Estimation (DACE)

In this section, we introduce how we use average causal effect 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.

Defect SCM Training. Based on the clustering results, we transform the embeddings of layout snippets to the cluster space and then build the SCM between candidate layout patterns and systematic defect. First, distance matrix $\mathbf{D} \in \mathbb{R}^{n \times k}$ is computed, whose (j, i)-th entry $[\mathbf{D}]_{j,i}$ denotes the distance of j-th embedding to the center of cluster i. Then the distance matrix are converted to a cluster membership matrix $\mathbf{P} \in \mathbb{R}^{n \times k}$ whose entries indicate the probability of each embedding belonging to each cluster as follows

$$[\mathbf{P}]_{j,i} = \frac{\exp\left(-\mathbf{D}_{j,i}/\tau\right)}{\sum_{i'} \exp\left(-\mathbf{D}_{j,i'}/\tau\right)}, \tag{5.6}$$

where τ is a temperature parameter, set as 0.1 in this work. The layout snippets closer to the cluster center have higher probabilities.

With all layout snippets represented in the form of membership vectors in $\mathbf{P}$, we model the SCM between candidate layout patterns and systematic defect with a multilayer perceptron (MLP) $\mathcal{M}$ to characterize their causal relationship. MLP as a neural network can be regarded as directed graphs with directed edges from a lower layer to the layer above. The final output is based on the hierarchy of interactions between lower level nodes.

Proposition 1. *Given an l-layer feedforward neural network $N(l_1, l_2, ..., l_n)$ where l_i is the set of neurons in layer i has a corresponding SCM $M(X, [l_1, l_2, ..., l_n], [f_1, f_2, ..., f_n], P(E))$, where l_1 is the input layer and l_n is the output layer. Corresponding to every l_i, f_i refers to the set of causal functions for neurons in layer i.*

Proof. In a feedfoward neural network, each layer neurons can be regarded as functions of neurons in its previous layer, i.e. $\forall i \in l\colon \forall l_{i_j} \in l_i\colon l_{i_j} = f_{i_j}(l_{i-1})$. The input layer l_1 can be assumed to be functions of exogenous variables E such that $l_{1_i} = f_{1_i}(e_i) \forall l_{1_i} \in l_1$ and $e_i \in E$. This structure in the random variables, neurons in the network, can be equivalently expressed by a SCM $M(X, [l_1, l_2, ..., l_n], [f_1, f_2, ..., f_n], P(E))$. □

The causal structure can be reduced to SCM $M^{'}(X, [l_1, l_n], f^{'}, P(E))$ by marginalizing out hidden neurons, since only the neurons in layer l_1 and layer l_n are observables.

Corollary 1. *Every l-layer feedfoward neural network $N(l_1, l_2, ..., l_n)$ with l_i denoting the set of neurons in layer i, has a corresponding SCM $M(X, [l_1, l_2, ..., l_n], [f_1, f_2, ..., f_n], P(E))$ which can be reduced to an SCM $M^{'}(X, [l_1, l_n], f^{'}, P(E))$.*

Proof. Starting with each neuron l_{n_i} in the output layer l_n, the corresponding causal function $f_{n_i}(l_{n-1})$ can be substituted as $f_{n_i}(f_{n-1_1}(l_{n-2}), f_{n-1_2}(l_{n-2}), f_{n-1_3}(l_{n-2}), ..., f_{n-1_{|l_{n-1}|}}(l_{n-2}))$. This can be written as $l_{n_i} = f^{'}_{n_i}(l_{n-2})$. f_{i_j} refers to the causal function

of neuron j in layer i and l_{i_j} refers to neuron j in layer i. Proceeding recursively layer by layer, we have modified functions such that, $\forall l_{n_i} \in l_n : l_{n_i} = f'_{n_i}(l_1)$. The causal mechanisms set f' of the reduced SCM M' would be $\{f'_{n_i} | l_{n_i} \in l_n\} \cup \{l_{1_i} = f_{l_i}(e_i) | l_{1_i} \in l_1$ and $e_i \in E\}$

□

With Proposition 1 and Corollary 1, we can simplify the SCM as $\mathcal{M}$ and concentrate on the input and output layers of $\mathcal{M}$ in the following procedures. The input layer l_1 of $\mathcal{M}$ has k input neurons x_i, $i \in \{1, \ldots, k\}$, each of which corresponds to one layout pattern. Its output layer l_n has one output neuron indicating the probability of systematic defect. The objective function for training the $\mathcal{M}$ is

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{e=1}^{m} \log \left[\sum_{i} p(r^e | y_i) p(y_i | \boldsymbol{\mu}_i, \boldsymbol{\theta}) \right], \tag{5.7}$$

where m is the number of diagnosis reports, $\boldsymbol{\theta}$ denotes the parameters of $\mathcal{M}$, $\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{j \in C_i}^{n} \mathbf{P}_{j,:}$ is the mean representation of cluster i with n_i layout snippets, $p(y_i | \boldsymbol{\mu}_i, \boldsymbol{\theta})$ is the output of $\mathcal{M}$ corresponding to layout pattern x_i, which indicates the probability of layout pattern x_i inducing the systematic defect, $p(r^e | y_i)$ is the conditional probability of diagnosis report r^e if layout pattern x_i occurs. We train the neural network $\mathcal{M}$ by minimizing the negative log-likelihood in Equation (5.7). The detail on estimating $p(r^e | y_i)$ is elaborated in Section 6.4.

LPA by ACE estimation. After the objective in Equa-

tion (5.7) converges, $\mathcal{M}$ is viewed as an SCM containing candidate layout patterns and systematic defect. We assume that the true root cause has the most significant average causal effect on the systematic defect. Therefore, average causal effects of input neurons (corresponding to layout patterns) on the output neuron (corresponding to the systematic defect) are estimated as the metric to identify root causes. Causal neural attribution in Equation (5.1) is adopted to compute the ACE of each layout patterns on the systematic defect.

When training the $\mathcal{M}$, the inputs are representations obtained from the membership matrix $\mathbf{P}$ whose entries are continuous values between $[0, 1]$. Since entries in $\mathbf{P}$ indicate the probability of a layout snippet belonging to a certain cluster, they have monotonic property, i.e., the closer (j, i)-th entry is to 1 (resp. 0), the higher (resp. lower) the probability of j-th layout snippet belonging to cluster i is. As a result, when regarding x_i as a binary variable, the ACE of x_i on y characterizes the causal effect of the presence of layout pattern x_i on the systematic defect. This ACE can be estimated as

$$ACE^{y}_{do(x_i)} = \left|\mathbb{E}[y|do(x_i = 0)] - \mathbb{E}[y|do(x_i = 1)]\right|. \tag{5.8}$$

The interventional expectation when x_i is intervened to 0 in Equation (5.8) can be estimated using Equation (5.1) as

$$\begin{aligned}&\mathbb{E}[y|do(x_i = 0)] \approx f'(\boldsymbol{\mu}_{i0}) + \\ &\frac{1}{2}\mathrm{tr}(\nabla^2 f'(\boldsymbol{\mu}_{i0})\mathbb{E}[(l_{in} - \boldsymbol{\mu}_{i0})(l_{in} - \boldsymbol{\mu}_{i0})^T|do(x_i = 0)]),\end{aligned} \tag{5.9}$$

where f' refers to the mapping from the input of $\mathcal{M}$ to its output y, the vector of interventional expectation $\boldsymbol{\mu}_{i0}$ is obtained by intervening the value of i-th entry of $\boldsymbol{\mu}_i$ to 0. Similar steps apply for the computation of the interventional expectation when x_i is intervened to 1. After obtaining the ACE of all layout patterns on systematic defect, we normalize them to form the root cause distribution of all candidate layout patterns as

$$p(x_i) = \frac{ACE^y_{do(x_i)}}{\sum_{i'} ACE^y_{do(x'_i)}}. \tag{5.10}$$

5.3.4 Inference flow

An overview of this unified framework and modules of DLSC and DACE are introduced in Section 5.3.2 and Section 5.3.3. Here we give a detailed explanation on the inference flow of the framework.

Pseudocode of the inference flow is presented in Algorithm 2, lines 1-7 correspond to the inference steps of DLSC and lines 8-12 represent the procedure of DACE. Firstly, layout snippets are transformed to embeddings in a latent space and clustered within each layer with the dedicated optimal cluster number k_d respectively. Layer-wise clustering is performed due to the consideration of process variance of different layers and efficiency. Secondly, the membership matrix of layout snippets in each layer are computed and different matrices from all layout layers are concatenated along axis zero to form $\mathbf{P}$. An SCM $\mathcal{M}$ is then learned using the information in diagnosis reports. Lastly, the

Algorithm 2 The Inference flow of the Framework.

Input: $R = \{r^e\}_{e=1}^{m}$ - a set of diagnosis reports, $S = \{\boldsymbol{s}_j\}_{j=1}^{n}$ - a set of layout snippets of all potential root causes;

Output: Root cause distribution

for $d = 1 \rightarrow L$ **do**
  for $j = 1 \rightarrow |S|$ **do**
    Encoder$(\mathbf{s}_j) \rightarrow \boldsymbol{z}_j$, $\forall j \in d$; ▷ Equation (5.5)
  end for
  Get optimal k_d with highest silhouette score;
  Compute distance matrix $\mathbf{D}_d$ using optimal k_d;
end for
Construct $\mathbf{D}$ by concatenating $\mathbf{D}_d$, $\forall d \in \{1, \ldots, L\}$;
Convert $\mathbf{D}$ to $\mathbf{P}$; ▷ Equation (5.6)
Train defect SCM $\mathcal{M}(\boldsymbol{\theta})$; ▷ Equation (5.7)
for $i = 1 \rightarrow k$ **do**
  Calculate ACE of clusters; ▷ Equation (5.8)
end for
return Root cause distribution; ▷ Equation (5.10)

Table 5.2 Notation on Diagnosis Report Features.

Feature	Description
rule_id	ID of the potential root cause
s_j^e	The score of potential root cause j in r^e
h_j^e	DFM hits of potential root cause j in r^e
v_j	DFM violations of potential root cause j
$\langle x_j, y_j \rangle$	Coordinate of potential root cause j
layer	Layer name of current potential root cause
type	Defect category of current potential root cause

average causal effect of each cluster on the systematic defect is estimated according to Equation (5.8). The root causes are identified based on the estimated ACE of all candidate layout patterns.

5.4 Experimental results

We evaluate the effectiveness of our proposed framework by testing its root cause identification accuracy on six noise-free datasets from six different layout designs, forty noisy datasets from five layout designs, and fifty mixture datasets from five layout designs. The advantage of our framework over compared methods in runtime is also validated by all related experiments. The accuracy is defined as the percentage of datasets that the real root cause is identified.

Table 5.3 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
Case 4	1091×1304	8	4176k
Case 5	2300×2410	9	5598k
Case 6	1483×1736	7	455k

5.4.1 Datasets

Encoder Training. We crop layout snippets according to the point of interests (POI) from the layout design in Case 2. Their size is determined by the pitch size in the corresponding layer. Layout snippets in other designs are not used during the training. We find out that the trained encoder can be applied to new designs directly without sacrificing clustering quality. Each cropped layout snippet is rotated, mirrored and shifted to generate positive samples for itself. Samples corresponding to different original layout snippets are deemed negative samples in contrastive learning.

LPA. We follow the same steps of defect injection performed in [3] to generate diagnosis reports. It requires around one to two hours to generate reports for one injection experiment. De-

Table 5.4 Defect Injection Statistics.

	#TotalInjections	#Open	#Bridge
Case 1	68	50	18
Case 2	107	72	35
Case 3	93	69	24
Case 4	44	28	16
Case 5	25	18	7
Case 6	39	28	11
Case 2 noise*	963	648	315
Case 3 noise	736	544	192
Case 4 noise	221	145	76
Case 5 noise	176	125	51
Case 6 noise	429	356	73

* We increase the number of injection experiments which is greater than the statistics presented in [10].

fects of type *Open* and *Bridge* are considered in the injection steps. The detailed information within diagnosis reports are listed in TABLE 6.1. Three classes of datasets are considered in our evaluation: (1) Noise-free dataset. Besides three different layout designs presented in [10], three more layout designs are used to construct noise-free datasets. Basic information of six designs is shown in TABLE 6.2. Case 2 to case 6 are five real silicon datasets that result from real in-production IC. Diagnosis reports in one noise-free dataset share one single true

root cause of systematic defect. The data of each layout design consists of `#TotalInjections` noise-free datasets and both open and bridge types are considered in our experiments (TABLE 6.3). (2) Noisy dataset. A certain percentage of diagnosis reports in the dataset share a single true root cause and the remaining diagnosis reports have root causes different from the true one (i.e., noise). Different percentages of noise are considered during the process of injections and the sources of noise are randomly sampled from the entire layout designs among all metal layers. (3) Mixture dataset. Diagnosis reports in this dataset are divided into four portions. Each portion share one root cause independently. Three portions of root causes are true and the rest portion is different from the true one. E.g., proportion '50-20-20-10' in TABLE 6.6 means 50%, 20%, and 20% of diagnosis reports have three true root causes correspondingly and 10% of diagnosis reports have random noise.

Besides the diagnosis reports, layout snippets of potential root causes in these reports are another required inputs of our framework. Note that in the noisy dataset, both the number of injection and types of defect are the same across different noise levels. We merged them as 'Case # noise' in TABLE 6.3.

5.4.2 Implementation Details

The proposed framework is implemented in Python with PyTorch library [65]. The encoder network is trained using four

Table 5.5 Accuracy(%) on Noise-free Datasets.

Dataset	Baseline	Commercial Tool	LPA-DCE [10]	eLPA-DCE
Case 1	25.00	98.53	100.00	100.00
Case 2	55.88	92.52	98.04	98.04
Case 3	58.06	98.92	98.92	100.00
Case 4	43.71	72.09	97.67	100.00
Case 5	39.25	82.05	91.3	100.00
Case 6	56.29	84.62	94.87	89.74
Average	46.37	88.12	96.8	**97.96**

Nvidia Tesla V100 GPUs. SGD optimizer is adopted with initial learning rate 1e−1, weight decay 5e−4, and momentum 0.9. The batch size, number of epochs, margin $marg$ in Equation (5.3) are set to 64, 16, and 1.5 in the experiments, respectively. Regularization weights α and β in Equation (5.5) are set as 1 in all experiments. Following [14], we add a single linear layer before the output of encoder during training to avoid the feature collapsion problem.

When conducting LPA, the defect SCM $\mathcal{M}$ is trained using SGD optimizer with initial learning rate 1e−2, weight decay 1e−3, and momentum 0.9. The maximum number of epoch of model training is set as 100 and the training will be early stopped if there is no improvement of the training loss in consecutive 5 epochs. The conditional probability of diagnosis report r^e if

layout pattern x_i is true is calculated as

$$p(r^e|y_i) = \frac{h^e_{j*} s^e_{j*}}{v_{j*}} \mathbb{1}_{\{s|s\geq 90\}}(s^e_{j*}), \tag{5.11}$$

where $j* = \arg\max_{j\in C_i} \sum_e s^e_j \mathbb{1}_{\{s|s\geq 90\}}(s^e_j)$, $\mathbb{1}_{\{s|s\geq 90\}}(s^e_j)$ is an indicator function which evaluates to 1 if $s^e_j \geq 90$ and 0 otherwise.

5.4.3 Results and Analysis

We compare the proposed framework `eLPA-DCE` with `LPA-DCE` [10] and an industry-leading commercial tool. One Nvidia Tesla V100 GPU is used for inference. The DFM hits are the number of a potential root cause appearing in the diagnosis report, more DFM hits indicate the layout snippet is more likely to be the root cause to a certain extent. To justify the necessity of our causality-based approach, a diagnosis statistical approach is presented as the baseline. The baseline approach finds the root cause in the following steps: (1) Given a volume of diagnosis reports, collect the DFM hits and DFM violations of potential root causes. (2) Calculate the ratio between DFM hits and DFM violations of layout snippets and get mean ratio within each cluster. (3) The cluster with the top rank of ratio is regarded as the root cause predicted by the diagnosis reports.

On noise-free datasets. As shown in TABLE 6.4, our method outperforms the commercial tool 9.84% on average under the setting of noise-free defect injection. The average accuracy of the baseline is 46.37% which is much lower than our method and the commerical tool. This indicates that by using

simple statistics according to the diagnosis report it is hard to locate the root cause accurately. With the new clustering loss, we got 1.16% improvement compared to `LPA-DCE` [10]. This means we get a better quality on membership matrix with the proposed regularization.

On noisy datasets. When given more challenging tasks of root cause identification, we observe that the performance of baseline becomes worse when the noise level is higher. Our proposed method can estimate the root cause with better performance under different noise levels, see TABLE 6.5. The average accuracy of the framework is 40.01% higher than the commercial tool. In case 5, the commercial tool can not identify the root cause under the noise perturbation. Our method can identify the root cause with robust performance across different noise levels. For other cases, especially when the ratio of noise is greater than 70%, it is difficult for the commercial tool to identify the root cause precisely. While our framework locates the root cause with higher accuracy than the commercial tool. E.g., the performance of case 3 and case 4 of the commercial tool at 80% noise level is 26.88% and 14.29% and ours are 100% and 76%. The proposed method is robust to the injection noise and it also outperforms `LPA-DCE` [10] in most cases.

On mixture datasets. We conduct the mixture root causes identification experiments to test whether our framework can be extended to multiple root causes scenario. The experimental results in TABLE 6.6 show that the proposed framework

has competitive results compared to the commercial tool especially in tasks of identifying three true root causes from mixture dataset, while the commercial tool may provide misleading results which can not be used for further study. Especially when the percentage of root causes is low, e.g., proportion '20-20-20-40', the causal-based methods are much more reliable, since it is more frequent we have a low percentage of root causes. Similar to the results of the commercial tool presented in TABLE 6.5, the root causes of case 5 can not be identified. We infer that the pitch sizes of patterns used in the commercial tool are different and this might be the reason why the identification of root causes of this case failed. Our framework achieves 82.91% high accuracy and got 47.17% better on average than the commercial tool. `eLPA-DCE` also got better results on four out of five designs than the `LPA-DCE` [10].

Top-3 accuracy on mixture datasets. We analyze the performance of identifying 1, 2, and 3 true root causes from the mixture datasets. Since there are 3 true root causes in each dataset, a true root cause belonging to one of top-3 layout pattern in the root cause distribution is a successful identification.

The accuracies of identifying 1, 2, and 3 true root causes are shown in Tables 5.8 to 5.12, our method can identify at least 1 root cause in all cases while the commercial tool fails to achieve 100% accuracy. Also, the average accuracy of identifying 2 root causes of our method is greater than 95%, which is around 30% better than the commercial tool.

Comparing the performance of `eLPA-DCE` and `LPA-DCE` [10], the root cause distributions generated with the encoder trained by clustering loss are more concentrated than the ones trained with conventional contrastive loss. Dense and recognizable clusters are beneficial to identify the root causes especially the number of clusters is large.

Inference speed. The inference time of our framework and the commercial tool on single root cause datasets are shown in Fig. 5.11. The speed of the proposed deep learning framework method surpasses the conventional commercial tool by a large margin. We got around $\times 10.4$ and $\times 2.3$ speedup on noise-free datasets and noisy datasets. The robustness of accuracy and inference speed indicate our method is valuable for industry.

Clustering quality. We compare the clustering quality of DLSC with directly applying k-means algorithm and DBSCAN algorithm on raw layout snippets using the adjusted rand index (ARI $\in [-1, 1]$, [80]). ARI computes a similarity measure between two partitions by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. High ARI indicates good match between the clustering results and the ground truth. In layout pattern analysis, high ARI scores indicate high resolution. The experiments are conducted on case 2 to case 6, 256 layout snippets are sampled layer-wise of each case for evaluation. Note that these cases are not available during the training of the encoder network. The ARIs of clustering using raw lay-

out snippets, embeddings in LPA-DCE [10] and embeddings in this work presented in Fig. 5.12 is the mean of ten independent evaluations. The ARI scores of embeddings with the clustering loss outperform the conventional contrastive learning method presented in [10] in all metal layers. Also, both the embedding-based clustering get higher ARI scores than the raw-based clustering. This indicates the contrastive learning can improve the quality of layout pattern matching. The clustering loss can make it performs better since it avoids the drawback of the primitive method.

Advantage of clustering loss on encoder training. The number of layout snippets used to train encoder with contrastive learning proposed in LPA-DCE and the clustering loss presented in this work is shown in Fig. 5.13. Only 4×10^5 samples are required to train the encoder in this work, which is $10\times$ lower than the number of samples used in LPA-DCE [10]. The budget for training data and computation resources is much less. Regularization with clustering loss lower the requirements of training set scales and speed up the procedure of model training, which reduces budgets of retraining on new designs with different technological process.

□ **End of chapter.**

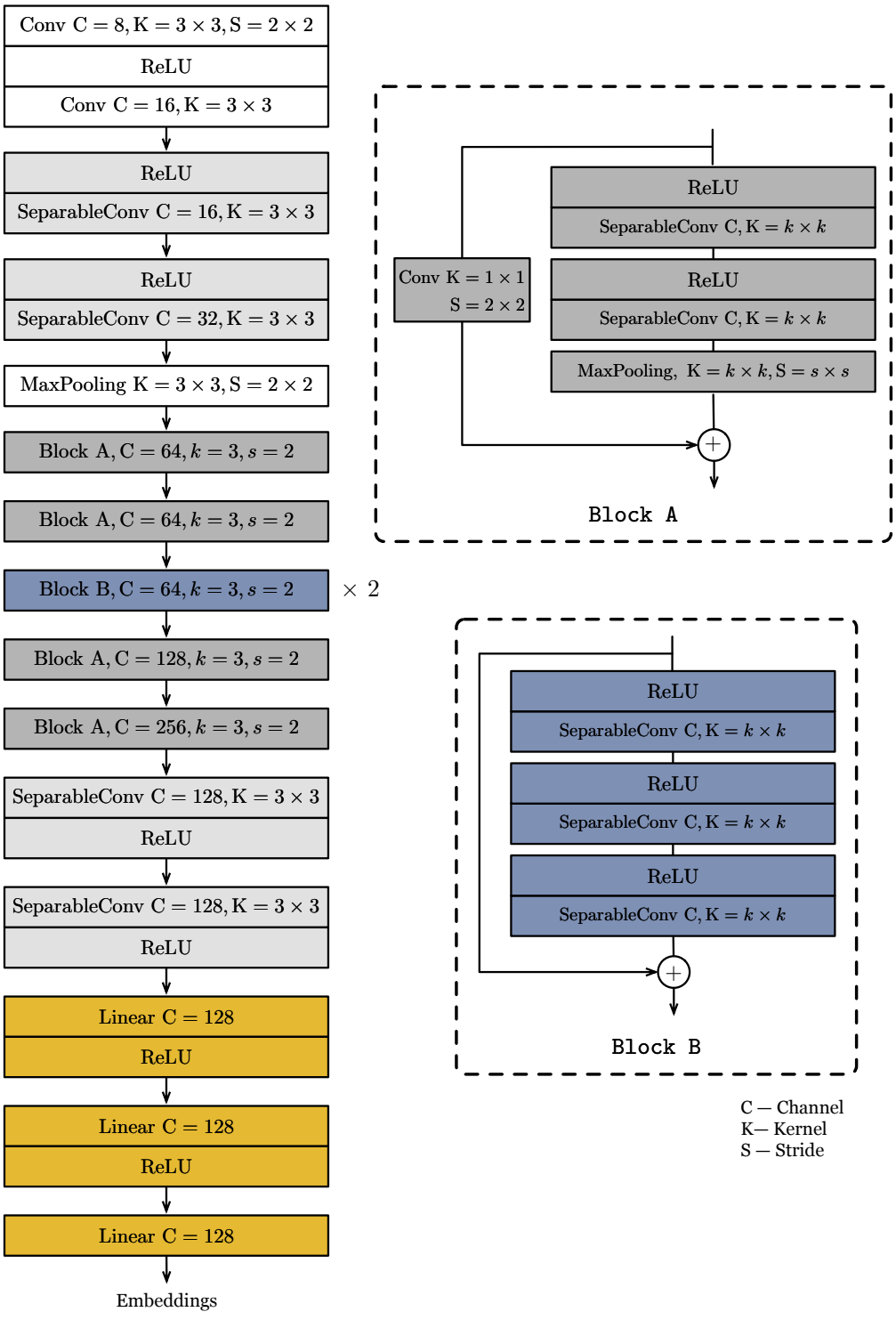


Figure 5.5 Encoder network structure for contrastive learning. Note that all Convolution, Separable Convolution and Linear layers are followed by batch normalization.

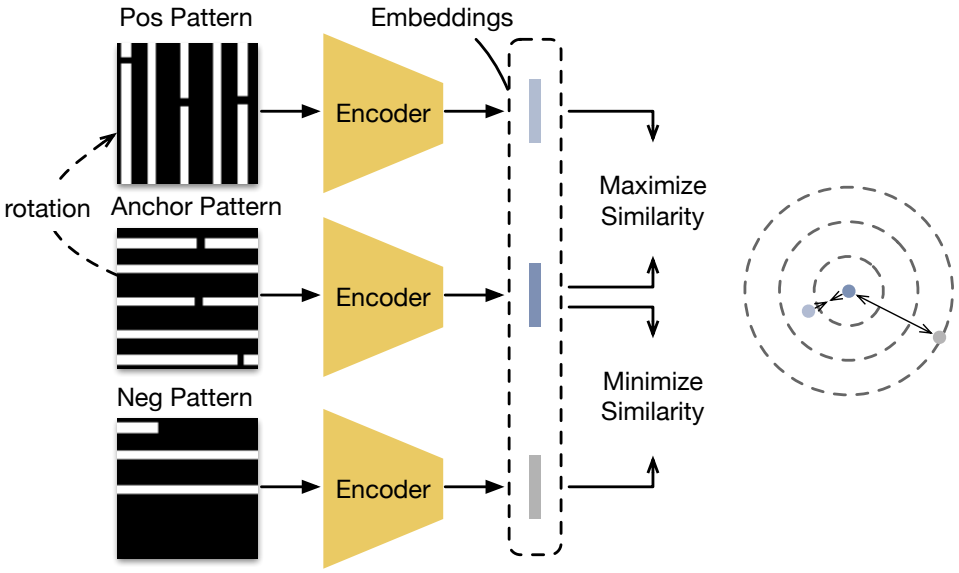


Figure 5.6 Illustration of encoder network training using contrastive learning. Parameters of encoders are shared.

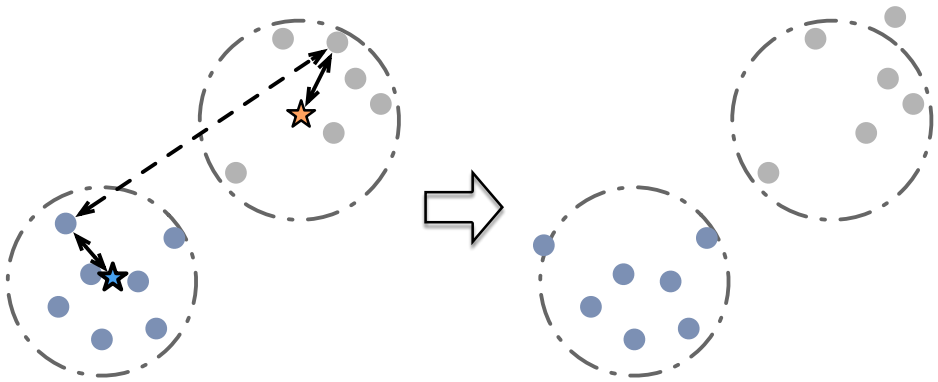


Figure 5.7 The drawback of contrastive loss. Pushing away two samples from different clusters may lead to negative effects on resolution.

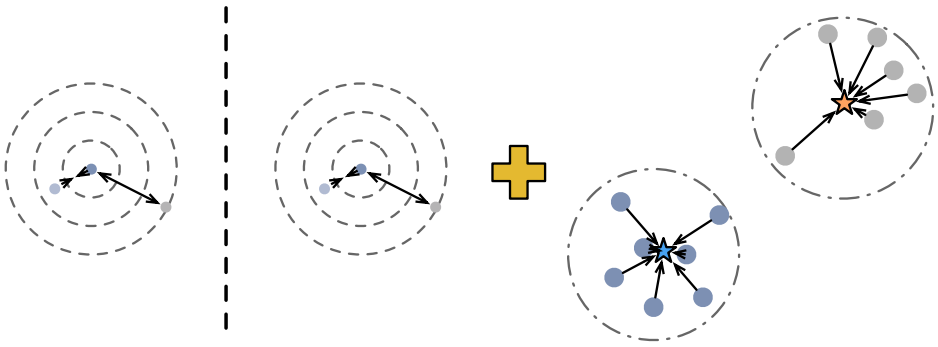


Figure 5.8 Contrastive Loss in [10] vs. Clustering Loss. The bold stars are the arithmetic mean centers of clusters.

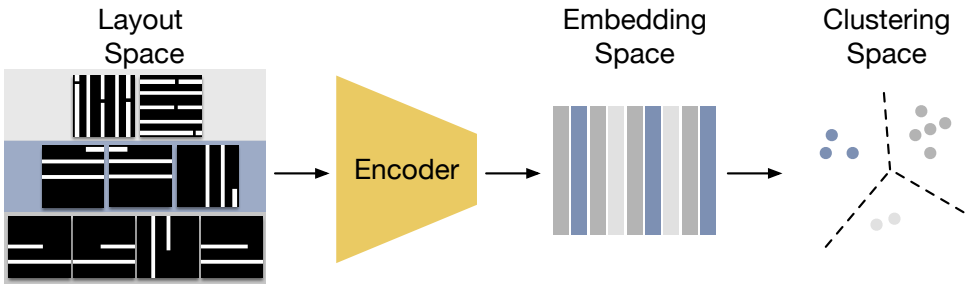


Figure 5.9 An example of Deep Layout Snippet Clustering.

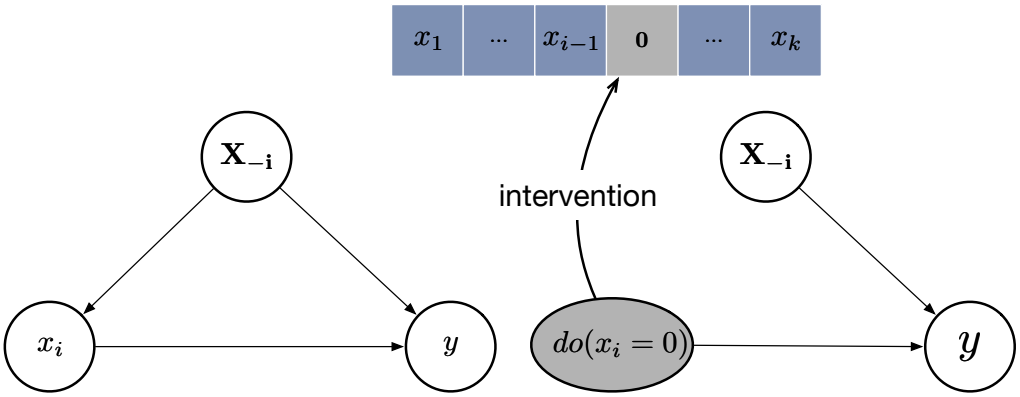


Figure 5.10 Left: The defect SCM for Layout Pattern Analysis without intervention. Right: Apply intervention on cluster i.

Table 5.6 Accuracy(%) on Noisy Datasets.

Noise(%)	Commercial Tool					LPA-DCE [10]					eLPA-DCE				
	Case 2	Case 3	Case 4	Case 5	Case 6	Case 2	Case 3	Case 4	Case 5	Case 6	Case 2	Case 3	Case 4	Case 5	Case 6
80	70.09	26.88	14.29	-	17.95	94.32	95.7	64	100	86.67	96.59	100	76	100	75.86
70	88.79	44.09	35.71	-	28.21	97.17	92.47	94.59	92	92.31	98.11	96.77	94.59	96	89.74
60	93.46	52.69	50	-	46.15	92.52	97.85	95.12	92	94.87	93.46	97.85	95.12	92	89.74
50	93.46	69.89	52.38	-	64.1	87.85	98.92	88.1	96	89.74	91.59	97.85	95.24	92	87.18
40	93.46	89.25	57.14	-	82.05	87.85	95.7	95.24	100	89.74	89.72	97.85	95.24	95.24	79.49
30	93.46	90.22	61.9	-	74.19	91.59	94.57	100	89.74	96.77	86.92	94.57	97.62	84.21	90.32
20	92.52	90.11	64.29	-	78.57	94.39	98.9	97.62	100	92.86	99.07	98.9	100	94.44	92.86
10	93.46	97.73	66.67	-	84.62	99.07	100	100	94.44	100	99.07	100	100	100	92.31
Average	89.84	70.11	50.30	-	59.48	93.10	96.76	91.83	**95.49**	**92.87**	**94.32**	**97.97**	**94.23**	94.24	87.19

Table 5.7 Accuracy(%) on Mixture Datasets.

Proportion (r1%-r2%-r3%-noise%)	Commercial Tool					LPA-DCE [10]					eLPA-DCE				
	Case 2	Case 3	Case 4	Case5	Case6	Case 2	Case 3	Case 4	Case5	Case6	Case 2	Case 3	Case 4	Case5	Case6
20-20-20-40	62.07	8.05	29.21	-	13.83	77.19	78.02	64	79.37	86.32	85.96	93.41	84	74.6	87.37
30-20-20-30	65.91	19.28	29.03	-	19.57	82.35	89.01	82.61	72.6	83.15	80.88	92.31	78.26	79.45	85.39
30-30-20-20	71.74	38.82	32.22	-	41.94	80.23	86.81	78.26	84.34	91.11	93.02	92.31	69.57	84.34	88.89
30-30-30-10	79.17	74.42	40	-	45.83	89.47	93.41	92.31	87.88	87	86.32	96.7	88.46	87.88	92
40-20-20-20	70.79	27.06	30	-	38.71	84.51	85.71	69.23	81.43	91.11	84.51	91.21	69.23	74.29	84.44
40-30-20-10	69.15	62.79	35.23	-	45.45	82.76	83.52	84.09	92.05	91.3	88.51	98.9	84.09	95.45	92.39
40-30-30-0	83.51	81.32	42.42	-	55	91.84	96.7	89.19	89	87	92.85	96.7	86.49	89	95
50-20-20-10	70.65	61.63	31.33	-	32.63	72.46	90.11	78.79	70.27	89.16	81.16	91.21	75.76	70.27	91.57
50-30-20-0	73.12	84.27	40.22	-	46	83.13	91.21	90.24	83.7	95.45	85.54	96.7	80.49	92.39	98.86
60-20-20-0	72.22	74.71	37.08	-	37.11	81.54	82.42	83.78	68.57	86.9	78.46	89.01	81.08	81.43	94.05
Average	71.83	53.24	34.67	-	37.61	82.55	87.69	**81.25**	80.92	88.85	**85.72**	**93.85**	79.74	**82.91**	**91.00**

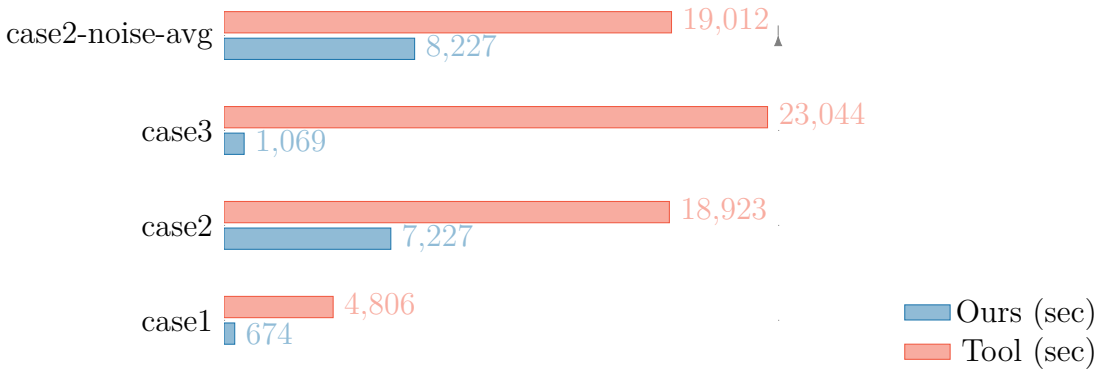


Figure 5.11 Inference speed comparison between our framework and the commercial tool.

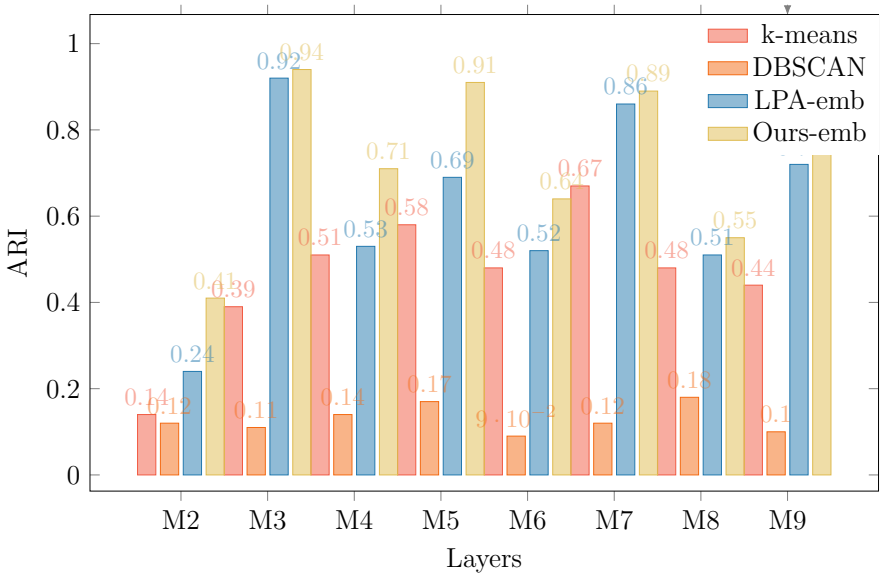
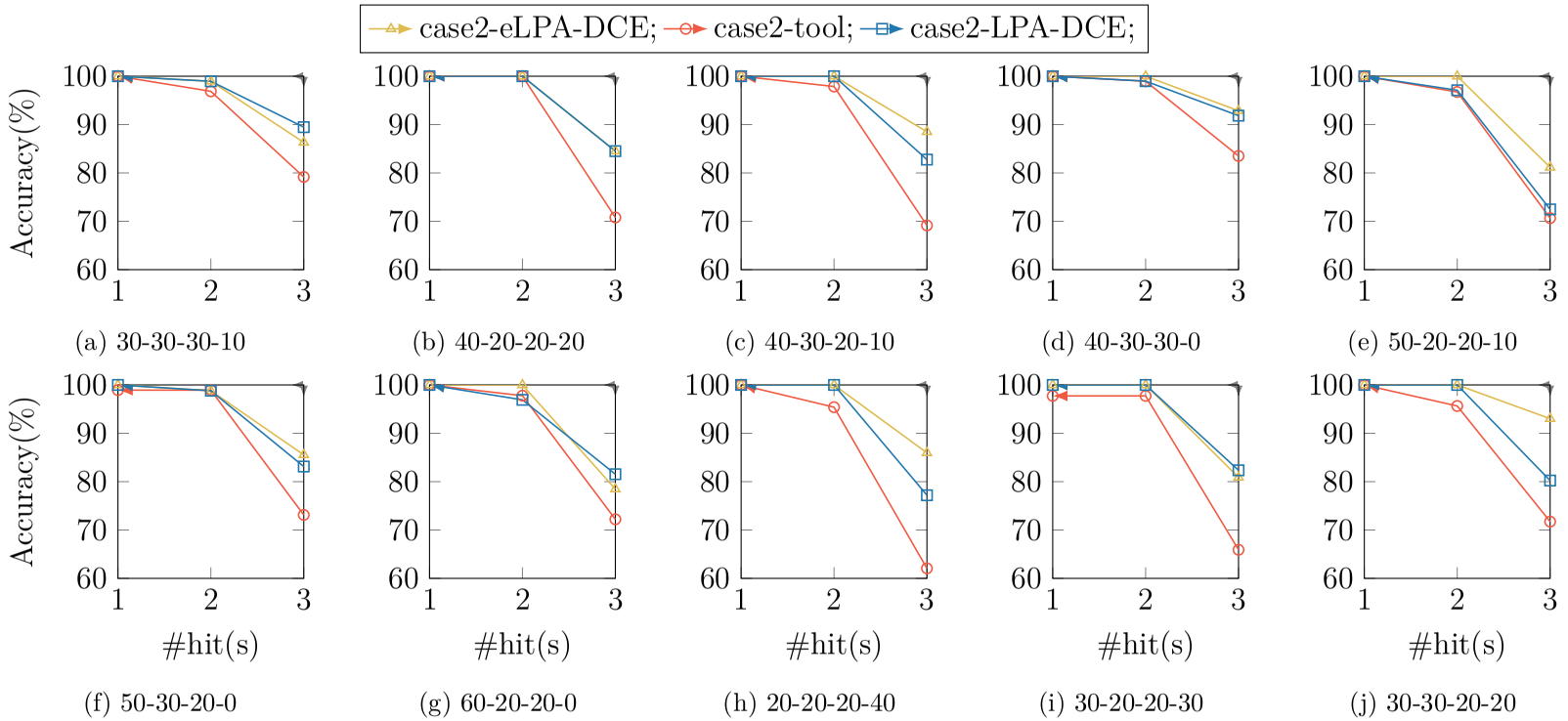


Figure 5.12 ARI of conducting layout pattern matching using raw layout snippets (k-means and DBSCAN), embeddings presented in `LPA-DCE`(LPA-emb) [10] and embeddings presented in this work(Ours-emb).



(a) 30-30-30-10 (b) 40-20-20-20 (c) 40-30-20-10 (d) 40-30-30-0 (e) 50-20-20-10

(f) 50-30-20-0 (g) 60-20-20-0 (h) 20-20-20-40 (i) 30-20-20-30 (j) 30-30-20-20

Table 5.8 Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture dataset case 2.

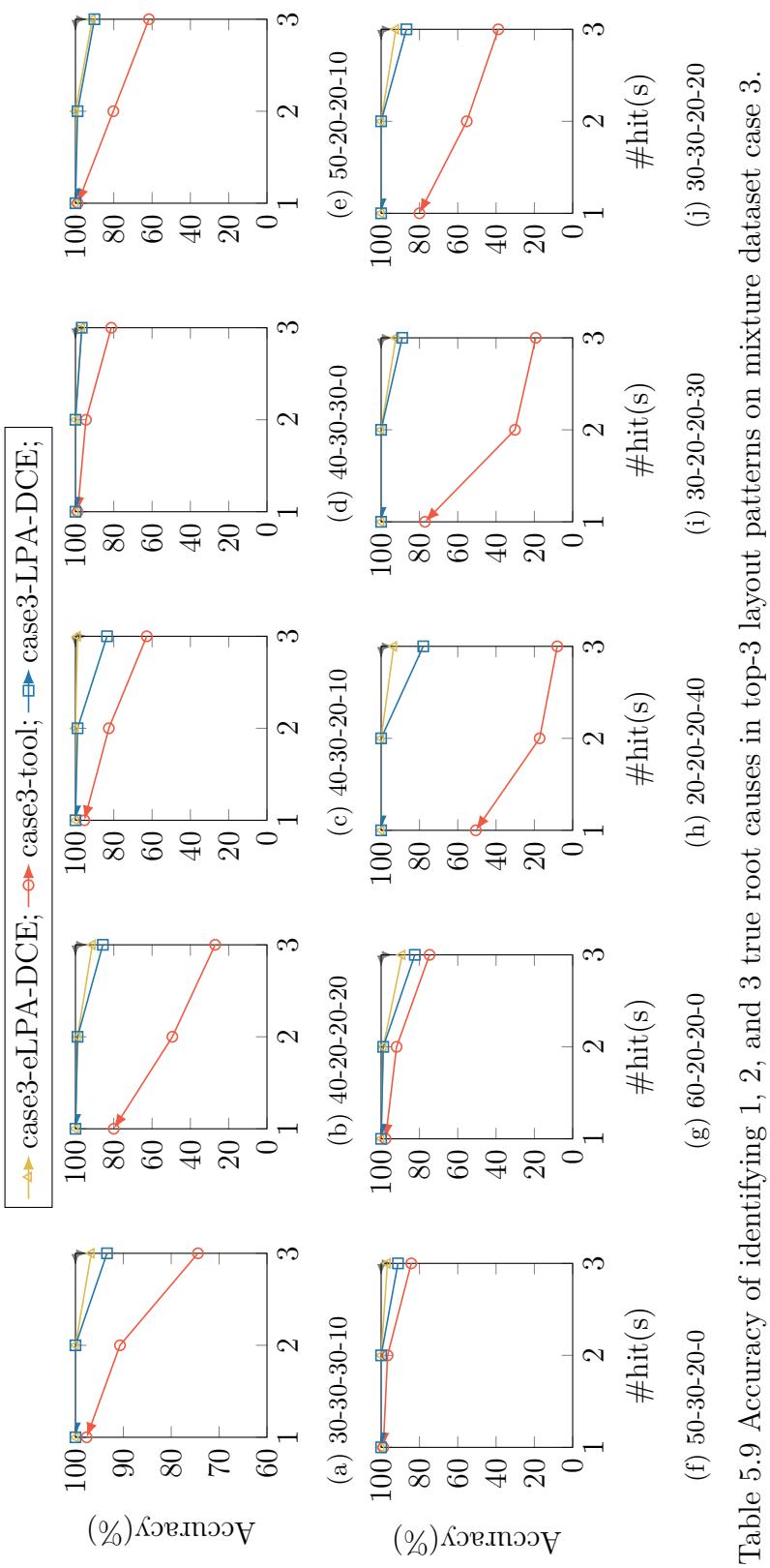


Table 5.9 Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture dataset case 3.

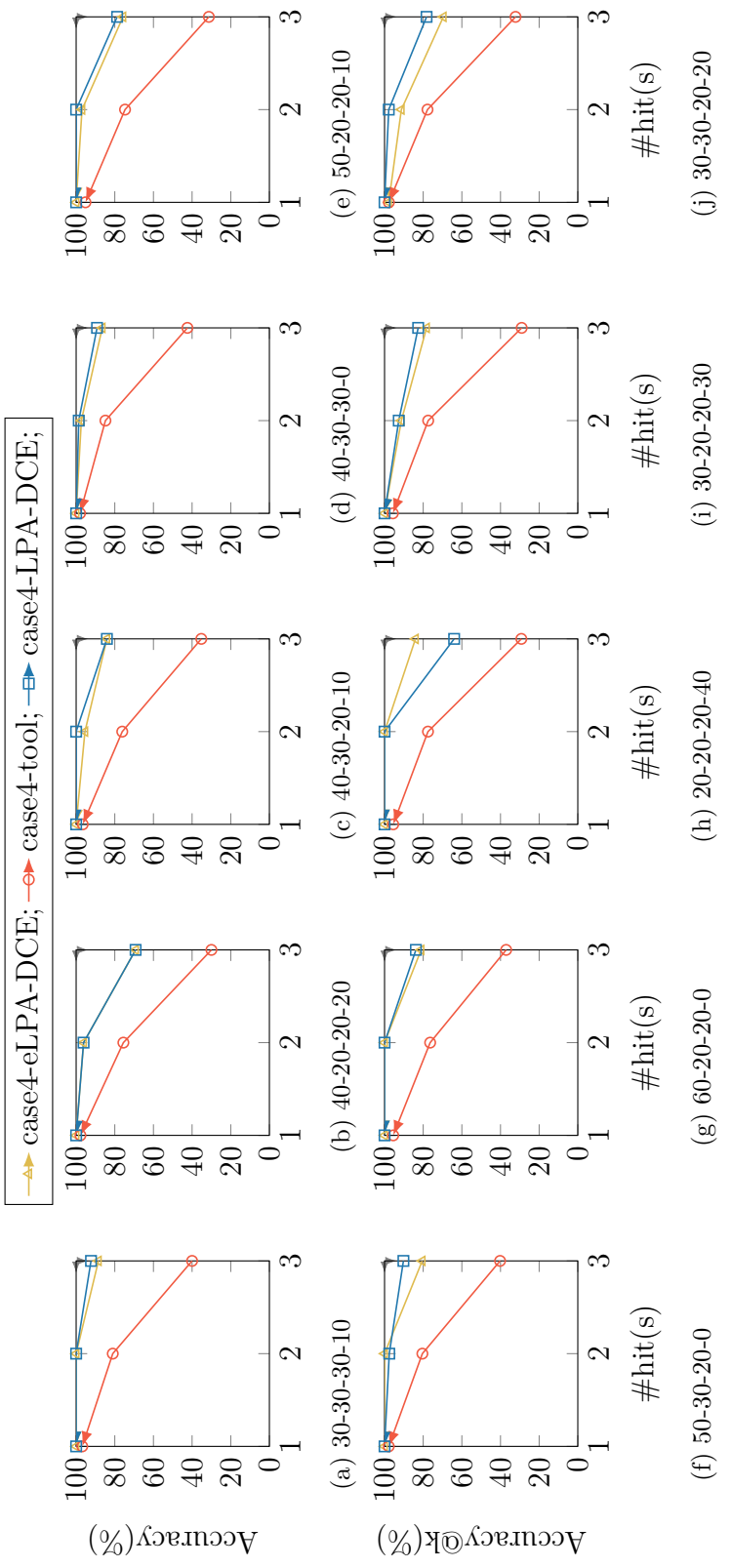


Table 5.10 Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture dataset case 4.

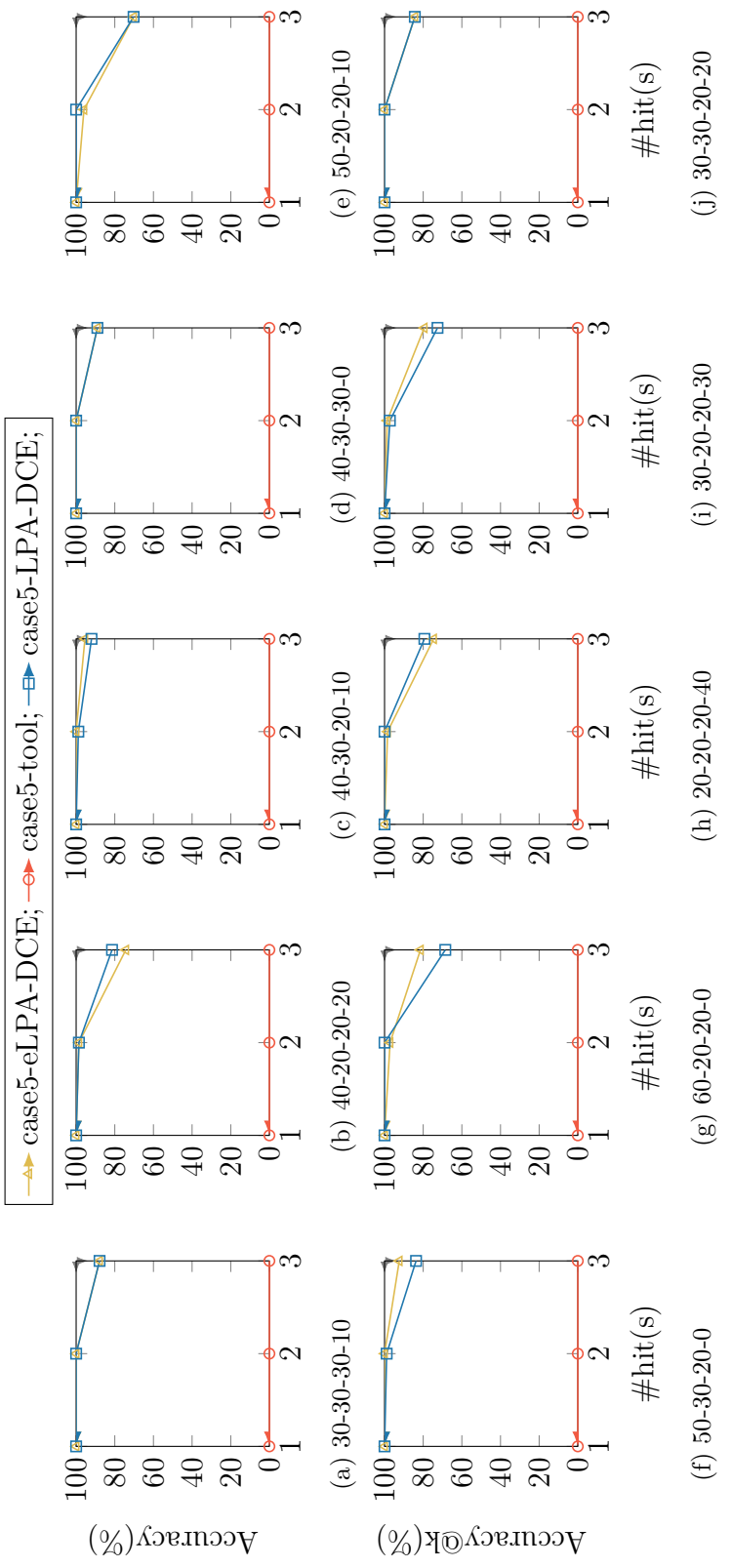
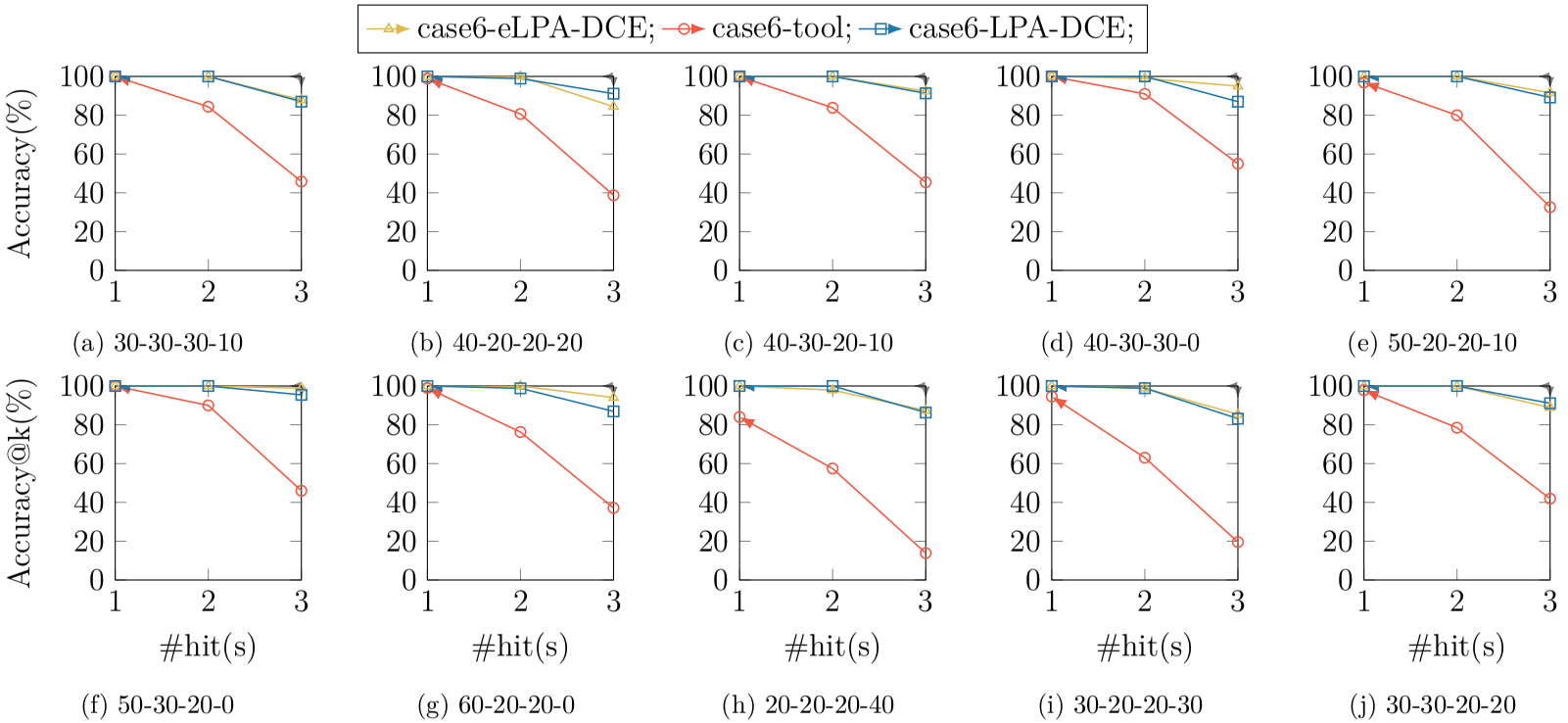


Table 5.11 Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture dataset case 5.



(a) 30-30-30-10 (b) 40-20-20-20 (c) 40-30-20-10 (d) 40-30-30-0 (e) 50-20-20-10

(f) 50-30-20-0 (g) 60-20-20-0 (h) 20-20-20-40 (i) 30-20-20-30 (j) 30-30-20-20

Table 5.12 Accuracy of identifying 1, 2, and 3 true root causes in top-3 layout patterns on mixture dataset case 6.

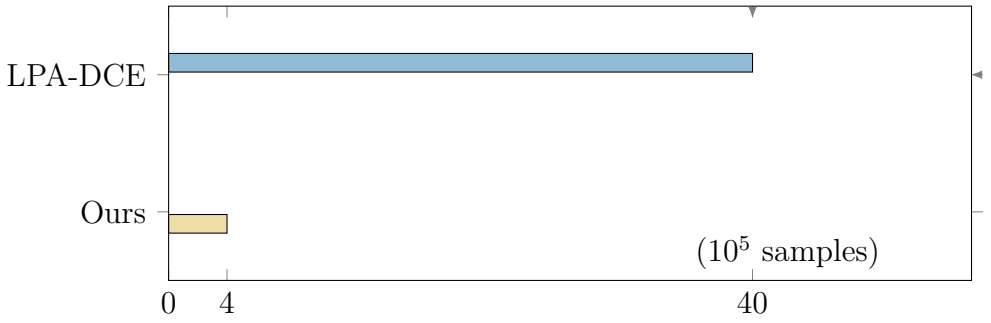


Figure 5.13 Training sample requirements for training encoder with contrastive learning proposed in `LPA-DCE` vs. the clustering loss used in this work.

Chapter 6

Root Cause Identification via Reinforcement Learning

6.1 Introduction

With the feature size shink down, specific layout patterns that are hard to fabricate tend to cause more systematic defects, such as open or bridge defects in neighboring wires. These layout patterns are an important source of yield loss. The yield of semiconductor devices refers to the percentage of non-defective dies of all dies manufactured. To ensure the profitability and reliability of products, a high and stable yield is expected to be achieved before volume production. Since layout configurations of new designs may differ from existing ones, identifying layout patterns that lead to yield loss through test chips, SRAMs, etc., are becoming less effective. Traditional yield learning methods such as Physical failure analysis (PFA), test structure, and inline inspection have been widely used to identify the root causes.

However, these methods are usually less effective due to the design iteration process and require both experience and a proper understanding of the fabrication process. A reliable PFA result documentation methodology is required to detect the design commonality across a large number of samples over a long period. Detecting defects [6,12,32] on entire layouts may adversely affect chip area and performance due to the overcorrection.

Many researchers consider the property of layout features. Clustering and pattern matching methods are adopted to help to improve systematic defect identification. Such as centroid-based and connectivity-based clustering methods presented in [85,86], layout snippets sharing the equivalent geometric structure are grouped. These methods have limitations on improving resolution, because failure analysis experts still need to judge whether the layout snippets in clusters are critical or not in simulation experiments. And the layout snippets that are shift-equivalent are regarded as different candidates, which further increases the difficulty.

A separate branch of works identifies the root causes of systematic defects using statistical methods. Such methods identify critical physical defects by analyzing volume diagnosis reports. [18,87] tried to improve the root cause identification accuracy by evaluating the impact of diagnosis features. Monte-Carlo simulation is presented in [87] to explain the impact and feasibility of factors in diagnosis reports. To speed up the processing time on identifying the systematic defects, [18] used data

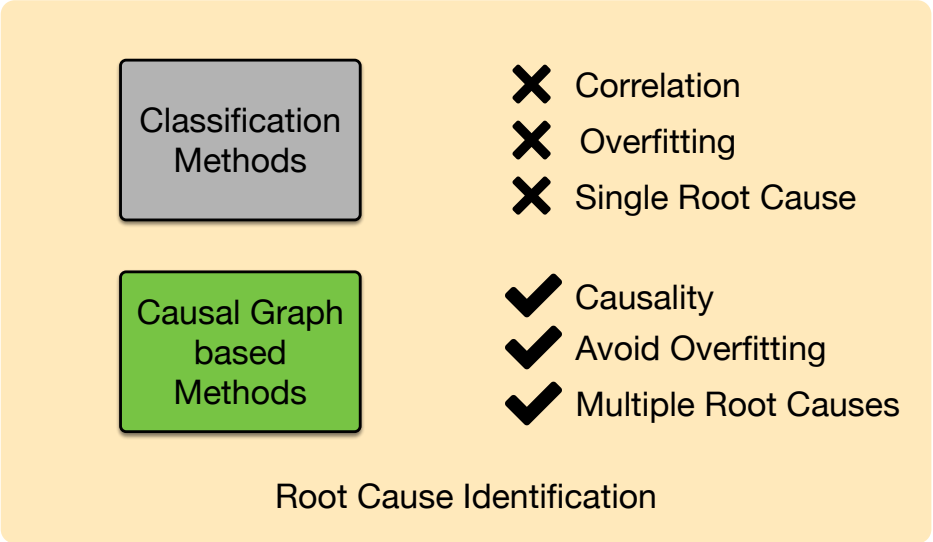


Figure 6.1 A summary on the advantage of Causal Graph based methods.

mining methods to reduce the PFA cost. The Bayesian method is used in [3] identifying the root causes of systematic defects based on the conditional distribution, which is a pioneering work in the field of determining root causes. By maximizing the likelihood of observed diagnosis report with the Expectation-Maximization (EM) learning algorithm, the optimal root cause distribution can be inferred. While these works focus on the diagnosis reports and overlook the layout patterns which limit the applicability in real scenarios.

Recently, several works take advantage of deep learning technology to further improve the identification quality. A self-adaptive deep learning framework is introduced in [42] to analyze the root cause. Compared to the conventional EM algorithm, the proposed classification method can transfer to new designs efficiently. The layout features are not considered in

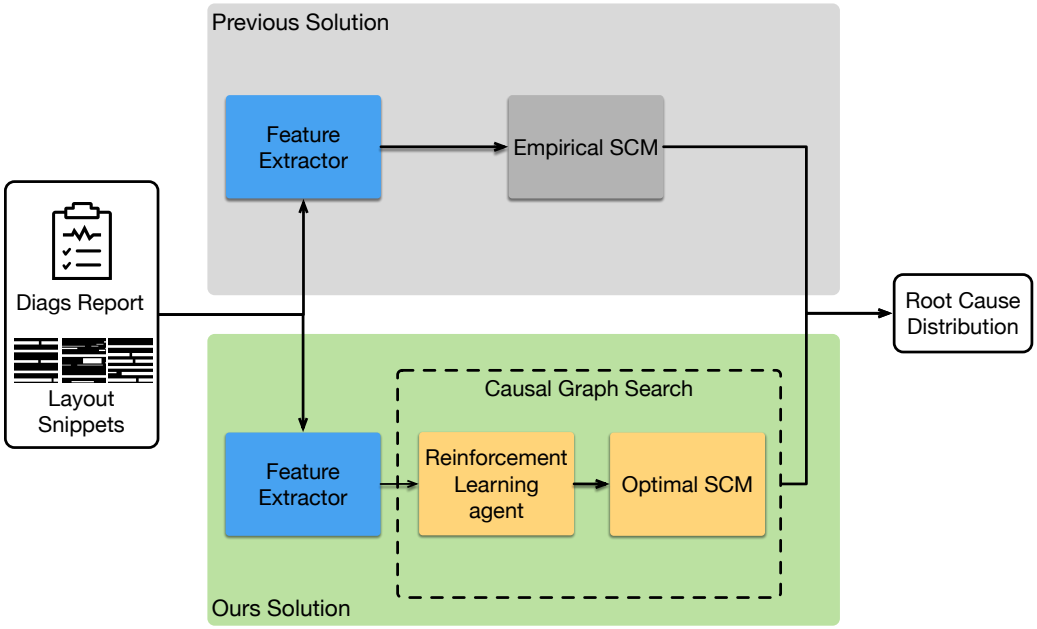


Figure 6.2 The proposed reinforcement learning based framework find the optimal causal relations which outperforms the empirical Structural Causal Model presented in [10].

this work. Chen *et al.* [10] presented a layout pattern analysis framework to predict the root causes via causal inference. Both layout patterns and diagnosis reports are utilized to infer the root cause distribution and provide high-resolution clusters for analysis, which provide a unified solution to this problem. Causal graphs are used in their work to construct causal relations. With the causal property, the overfitting problem is avoided, which is common in classification-based methods, such as [42]. Moreover, conventional classification methods are limited to identifying the only single root cause, whereas the proposed structure causal model can identify multiple root causes.

The reasons why causal graph-based methods outperform the classification methods are highlighted in Fig. 6.1.

One major drawback in [10] is that all the statistical features in diagnosis reports are included in the causal graph where some of these observed variables may bring uncertainty to the model. Especially when some physical features are biased or not precise in diagnosis reports, the empirical causal graph may not ensure the identification accuracy, since the confounders lead the incorrect causal relations. We observed that some features oriented to the high probability of root causes can not imply that they are the true causes. To tackle this problem, we propose a reinforcement learning-based framework to improve the root cause identification quality. Reinforcement learning has a principled way to choose problem-specific combinations of score functions and search strategies, which is an advantage to search for the optimal causal graph. With the optimal causal graph, we can locate the root cause more precisely. The advantage of our work compared to [10] is shown in Fig. 6.2. We use an encoder-decoder network as the actor-network. The layout features are encoded as latent codes to represent the states and the outputs of the decoder as the representation of the causal graphs. The action space is carefully designed to improve the robustness of searching, The average causal effect-based reward function is designed to guide the reinforcement learning system to find the optimal causal graph cooperating with the critic network. Then the critic network updates the state of the actor-network. The

contributions of this work are summarized as follows:

- To make sure the agent can find the causal graph efficiently, a diagnosis-aware action space design is proposed. Instead of traversing the whole action space of suspect patterns, it is much more effective to learn with the diagnosis-aware action space.
- An encoder-decoder network is designed to speed up the learning procedure. States of the actor-network are represented in latent space and are utilized to update the critic network.
- A novel reinforcement learning framework for root cause identification is proposed. With the guidance of the actor-critic algorithm, redundant suspect patterns presented in diagnosis reports are wiped out. Detect quality of root causes outperforms the state-of-the-artwork and industry solutions.

The remainder of this paper is organized as follows. Section 6.2 introduces terminologies and problem formulation related to this work. Section 6.3 describes the algorithmic details and reinforcement learning agent of our framework. Section 6.4 lists the experimental results, followed by the discussion and conclusion in Section 6.5.

6.2 Preliminaries

In this section, preliminary knowledge and related works on reinforcement learning and structural causal learning is abbreviated reviewed.

6.2.1 Reinforcement Learning

The idea of reinforcement learning (RL) is to have an agent interact with the environment in order to learn. The optimal policy for sequential decision-making problems is learned through trial and error in different fields, such as the natural, social, and engineering sciences. With the help of big data, powerful computation, i.e. Graphics Processing Unit (GPU), and advanced software and hardware co-design, The reinforcement learning field has been experiencing a rebirth in recent years.

A reinforcement learning agent interacts with an environment over time. RL problems can be formulated as Markov Decision Processes (MDPs), consisting of following elements: actions, states, state transition, and reward:

- Actions A: a finite set of actions taken by the agent.
- States S: a finite set of representations of possible states of the environment.
- State Transition P: given a state and an action, the probability distribution over next states.

- Reward R: the reward signal after state transition with an action.

At time step t, the agent receives a state s_t in a state space S. An action a_t is selected from an action space A with a policy $\pi(a_t|s_t)$. Normally, $\pi(a_t|s_t)$ is the agent's behavior, i.e., a mapping from state s_t to actions a_t according to the reward function $R(s, a)$. The agent receives a scalar reward r_t, and transitions to the next state s_{t+1} with the state transition probability $P(s_{t+1}|s_t, a_t)$. In an episodic scenario, the agent will continue until reach a terminal state then restarts.

Deep reinforcement learning [2, 27, 63] gets attention in recent years owing to the blooming of deep learning. The components of reinforcement learning such as value function, $\hat{v}(s;\theta)$ or $\hat{q}(s, a;\theta)$, policy $\pi(a|s;\theta)$, and state transition function and reward function. Here, the parameters θ are the weights of the deep learning model. The difference between deep RL and conventional RL is that stochastic gradient descent is used as a function approximator to update weight parameters in deep RL. There has been many works on applying reinforcement learning methods in EDA problems such as [38, 51, 56, 61] in floorplanning, placement and routing.

6.2.2 Structural Causal Learning

Causality is a generic relationship between an effect and the cause that gives rise to it. The definition of causality is hard to

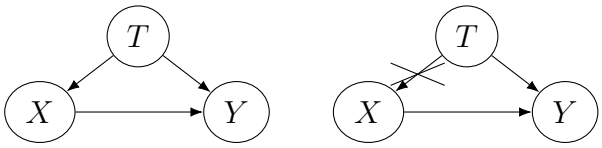
describe since the causes and effects we know about are intuitive in general. Different from the statistical associations, identifying the causation is the key step towards human-level intelligence and can serve as the foundation of artificial intelligence.

Structural Causal Model (SCM) is a conceptual model that describes the causal mechanism of a system. An SCM normally consists of the causal graph and the structural equations. A causal graph forms a special class of Bayesian network with edges representing the causal effect.

Definition 5 (Structural Causal Model [67])**.** *A structural causal model M is a 4-tuple $(Ex, En, F, P(Ex))$, where*

- *Ex is a set of exogenous variables, which is unobserved. Its measure is determined outside the model;*
- *En is a set of endogenous variables, which is observable. Its measure is determined by the model;*
- *F represents a collection of functions $F = \{f_i\}$ such that each endogenous variable $x_i \in X$ is determined by a function $f_i \in F$, where f_i is a mapping from the respective domain of $\epsilon_i \cup Pa_i$ to x_i, with $\epsilon_i \subseteq E$, $Pa_i \subseteq X \backslash \{X_i\}$ is the set of direct parents of x_i;*
- *The uncertainty is encoded through a probability distribution over the exogenous variables, $P(Ex)$.*

SCMs provide a compact way of characterizing average causal effect $ACE^{y}_{do(x_i)}$, which is defined as $\mathbb{E}[y|do(x_i = 1)] - \mathbb{E}[y|do(x_i =$



(a) SCM without intervention.

(b) SCM with intervention

0)] for binary x_i. $\mathbb{E}[y|do(x_i = \alpha)]$, known as *interventional expectation* [67], denotes the expectation of y when intervening the value of x_i to be α. For an SCM, such intervened model can be represented by replacing the structural equation $x_i = f_i(Pa_i, \epsilon_i)$ by a constant $x_i = \alpha$. They enable the codification of the existing knowledge in diagrammatic and algebraic forms and consequently leverage data to estimate the answers to interventional and counterfactual questions.

6.2.3 Problem Overview

Given a dataset contains batches of diagnosis reports $R = \{r^n\}_{n=1}^m$ and layout snippets of potential root causes in these reports. We want to identify the causal relationships between independent defects in reports and find the true root cause(s) causing systematic defects in reports R. The geometric structure of layout snippets will be used to find the optimal structure causal model.

This problem can be formulated as to minimize a score function $Score(g)$ to each directed acyclic graphs (DAGs) g according to the diagnosis reports and layout snippets. Then search over the space of all DAGs for the best scoring:

$$\min_{g} \quad Score(g) \tag{6.1a}$$

$$\text{s.t.} \quad g \in \textbf{DAGs}. \tag{6.1b}$$

The DAG with the best scoring can represent the causal relation among the suspects and identify the root causes correctly.

6.3 Algorithm and Framework

The objective of this work is to acquire an effective method to find the causal graph among suspect patterns, where the obtained causal graph can represent the causal relationship between potential root causes and systematic defects. To model this problem using reinforcement learning (RL), there are the following challenges we need to solve:

- Discovering the causal relation among the suspects with RL requires a proper measurement of graph quality. An effective score function is required to evaluate the quality of the causal graph.
- Generally, there are hundreds of suspects present in diagnosis reports. The original action space to find the optimal DAG which can represent the causation between suspects and root causes is extremely large in the combinatorial optimization problem.
- The number of the layout snippets is large, an effective representation is required as the prior knowledge to update

the critic-network.

In this section, we will introduce how to formulate the score function with the average causal effect, and then we discuss how to construct an effective action space to find the causal graph, and how to design the reward function. Finally, an overview of the proposed reinforcement learning framework will be presented.

6.3.1 Score Function with the guidance of Average Causal Effect

Mining the casual structure from observed diagnosis reports and layout features via score-based methods such as the Bayesian Information Criterion [94] and the Bayesian Gaussian equivalent score [31] is time-consuming. A feasible score function $S(g)$ is required to judge the quality of DAGs which can find the causation between suspects and root causes efficiently. We use the average causal effect as an estimator on root cause distribution under the assumption that the true root cause has the most significant average causal effect on the systematic defect. The average causal effect on each independent defect x_i is calculated as:

$$\tau^{y}_{do(x_i)} = \left|\mathbb{E}[y|do(x_i = 0)] - \mathbb{E}[y|do(x_i = 1)]\right|. \tag{6.2}$$

The notation of $do(\cdot)$ is the do calculus which simulates physical interventions by deleting or modifying certain functions from the model, replacing them with a constant value, while keeping

the rest of the model unchanged. Here $do(x_i)$ represents the intervention on defect x_i. The average difference between potential outcomes are calculated under the binary treatments. $do(x_i = 0)$ indicates the do operation on x_i by modifying the variable to zero, which represents the situation that the defect x_i does not appear in the design. $do(x_i = 1)$ means the do operation on x_i by modifying the variable to one, which represents the situation that the defect x_i exists in the design. The root cause distribution of suspect layout patterns is formed by normalizing the ACE scores of all layout patterns on the systematic defect, which is expressed as

$$p(x_i) = \frac{\tau^y_{do(x_i)}}{\sum_{i'} \tau^y_{do(x'_i)}}, \tag{6.3}$$

By sorting the rank of root cause probability, a long-tailed distribution will be generated ideally. The value of root causes should be greater than the suspects. The score function with the guidance of average causal effect is defined as

$$S(g) = -\frac{\sum_{i \in \text{root causes}} \mathbb{I}_{rank}(i) \times p(x_i)}{N}, \tag{6.4}$$

where N is the number of root causes and $\mathbb{I}_{rank(i)}$ is an indicator function:

$$\mathbb{I}_{rank}(i) = \begin{cases} 1 & \text{if the rank of root cause } i \text{ is correct} \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

If the rank of root cause i is consistent with the ground truth, the value is one, otherwise, it is zero. The causal graph identifying

the root cause correctly has lower score values, and the causal graph that can not identify any of the root causes will get zero.

6.3.2 Action Space Design

The original action space of a d-node graph can be presented as a adjacency matrix $W \in \mathbb{R}^{d\times d}$. Let $\mathcal{B}(W) \in \{0,1\}^{d\times d}$ be the binary matrix such that $[\mathcal{B}(W)]_{ij} = 1 \Leftrightarrow w_{ij} \neq 0$ and zero otherwise. $\mathcal{B}(W)$ defines the adjacency matrix of a directed graph $G(W)$. The objective to learn causal graph among action space is expressed as

$$\min_{W\in\mathbb{R}^{d\times d}} \quad S(\mathcal{B}(W)) \tag{6.6a}$$

$$\text{s.t.} \quad G(\mathcal{B}(W)) \in \textbf{DAGs}, \tag{6.6b}$$

where $G(\mathcal{B}(W))$ is the d-node graph induced by the binary adjacency matrix and S is a score function. Using a deep neural network as the actor-network to generate such cases is difficult since a directed graph g with binary adjacency matrix W is acyclic if and only if the trace of e^W is equal to the number of the vertex. Adding this regularization as a term in reward requires hyperparameter tuning which is time-consuming. To simplify the learning schedule, an effective action space design considering the structure of diagnosis reports is proposed.

The hierarchical structure of diagnosis reports is show in Fig. 6.3. A diagnosis report contains three layers includes logic faults layer, defects layer, and root causes layer. Logic faults

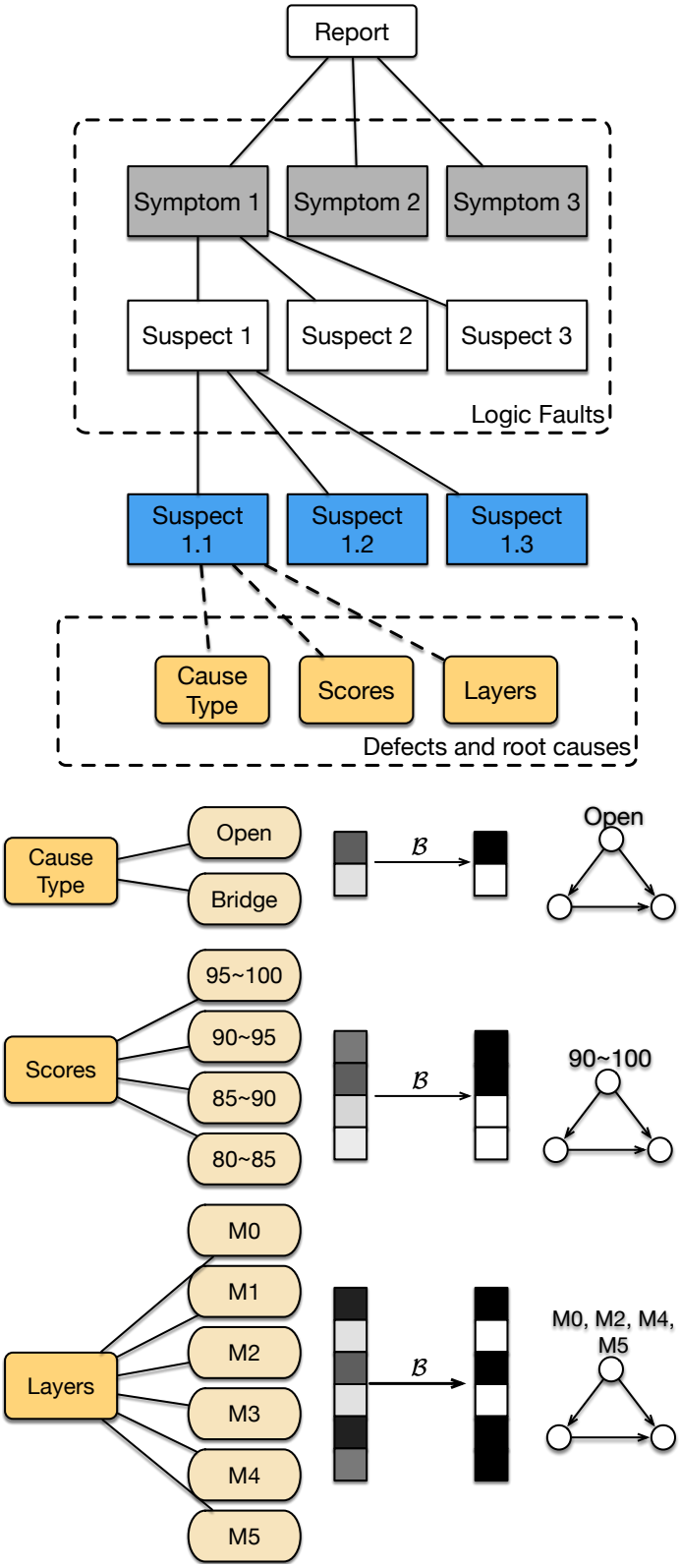


Figure 6.3 Top: Hierarchical structure of diagnosis reports. Bottom: Visualized illustration on action space design. The example report represents physical and logical features of a design including six layers, suspect scores ranging from 80 to 100, and two defect types.

are specific logic failures at a specific logic location in a netlist. Physical features such as cause types (e.g. open and bridge) and scores are included in the defect layer. Different defects are related to variance physical effects and at different physical locations. Cause types, scores, and layers are the most critical factors when judging the possibility of root causes in practice. Inspired by the structure of diagnosis reports and the objective of causal graph searching, the action space A is designed which includes three subsets:

$$A_{cause} := \{Open, Bridge\}, \tag{6.7a}$$

$$A_{score} := \{[80 + i * 5, 85 + i * 5] | i \in [0, 3], i \in \mathbb{Z}\}, \tag{6.7b}$$

$$A_{layer} := \{M_i | i \in [0, n), i \in \mathbb{Z}\}, \tag{6.7c}$$

where n is the number of metal layers of the design. Instead of predicting the binary adjacency matrix $\mathcal{B}(W)$, vectors $\boldsymbol{v}_s$, $\boldsymbol{v}_c$, and $\boldsymbol{v}_l$ are constructed to represent A_{score}, A_{cause}, and A_{layer} in our implementation. As shown in the bottom part of Fig. 6.3, the grey blocks indicate the predicted vectors $\boldsymbol{v}_s$, $\boldsymbol{v}_c$, and $\boldsymbol{v}_l$. Causal graphs are generated after the vectors are binarized with $\mathcal{B}$. The objective to find the causal graph among diagnosis based action space is expressed as

$$\min_{\boldsymbol{v}_s \in \mathbb{R}^{4\times 1}, \boldsymbol{v}_c \in \mathbb{R}^{2\times 1}, \boldsymbol{v}_l \in \mathbb{R}^{n\times 1}} \quad S(\mathcal{B}(\boldsymbol{v}_s)), S(\mathcal{B}(\boldsymbol{v}_c)), S(\mathcal{B}(\boldsymbol{v}_l)) \tag{6.8a}$$

$$\text{s.t.} \quad G(\mathcal{B}(\boldsymbol{v}_s)) \in \textbf{DAGs}, \tag{6.8b}$$

$$G(\mathcal{B}(\boldsymbol{v}_c)) \in \textbf{DAGs}, \tag{6.8c}$$

$$G(\mathcal{B}(\boldsymbol{v}_l)) \in \textbf{DAGs}. \tag{6.8d}$$

The agent learns among three action spaces and selects the optimal one as the target causal graph at inference. We use the softmax function with a constant threshold of 0.4 as $\mathcal{B}$ in this work.

6.3.3 Reward

The reward incorporates the score function in Equation (6.4) and the causal graph related constraints:

$$R(g, r) = -(S(g) + \lambda \mathbb{I}(r \notin \{A_{score}, A_{cause}, A_{layer}\})), \tag{6.9}$$

where r is the set of root causes and λ is the penalty weight. The indicator function $\mathbb{I}(r \notin \{A_{score}, A_{cause}, A_{layer}\}))$ will be one if the root causes are not included in predicted vectors $\boldsymbol{v}_s$, $\boldsymbol{v}_c$, and $\boldsymbol{v}_l$, e.g. one root cause is on $M1$, while the corresponding value is zero in $\boldsymbol{v_l}$. In this case, we can not identify this root cause by calculating the average causal effect. The probability of this root cause is zero. So the root causes must be included in the predicted causal graph or the reward should be penalized. The training objective of RL agent is expressed as:

$$Q(\boldsymbol{\theta}|\boldsymbol{x}) = \mathbb{E}_{A\sim\pi(\cdot|\boldsymbol{s})}[-(S(g) + \lambda \mathbb{I}(\boldsymbol{r} \notin \{A_{score}, A_{cause}, A_{layer}\}))], \tag{6.10}$$

where $\boldsymbol{x}$ are training samples drawing from the dataset. $\boldsymbol{\theta}$ and $\pi(\cdot|\boldsymbol{s})$ denote the model parameters and the policy.

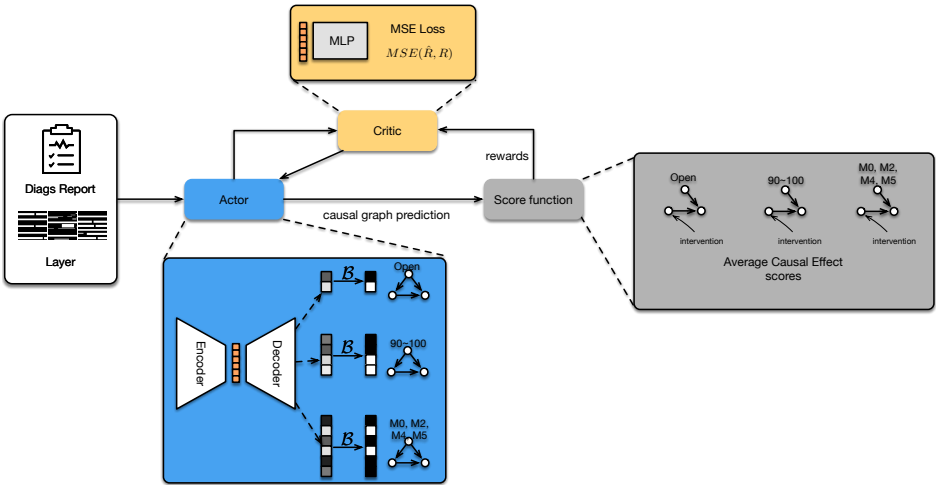


Figure 6.4 Reinforcement Learning Agent for Root Cause Identification. We use actor-critic algorithm as the guidance of exploitation and exploration scheme for causal graph learning.

6.3.4 RCI Reinforcement Learning Framework

By integrating the above reinforcement learning modules, the RCI reinforcement learning framework is constructed. A full view of the framework is shown in Fig. 6.4, the RL agent determines where to search automatically with the stochastic policy.

Actor-Network and Critic-Network. To facilitate the learning efficiency, an encoder-decoder based neural network is proposed as the actor-network and a three-layer Multilayer Perceptron (MLP) with a scalar output is used as the critic-network. The reasons why to use an encoder-decoder network to represent the actor-network are motivated by the following considerations:

- A simple and effective state representation is required for updating. The encoder is a feasible neural network struc-

Table 6.1 Notation on Diagnosis Report Features.

Feature	Description
s_j^n	The score of potential root cause j in r^n
h_j^n	DFM hits of potential root cause j in r^n
v_j	DFM violations of potential root cause j
layer	Layer name of current potential root cause
type	Defect category of current potential root cause

ture to transfer the origin features as encrypted codes.

- Sharing the embedding state codes with the critic-network is effective for training the actor and critic in synchronous.
- Different from the reinforcement learning frameworks solving floor planning or placement, generating the action space according to the corresponding state in this task is hard to model in an explicit way.

The diagnosis reports statistic features are encoded as the latent codes to represent the states of actor and then are decoded to generate $\boldsymbol{v_s}$, $\boldsymbol{v_c}$, and $\boldsymbol{v_l}$. The structure of the actor-network are presented in Fig. 6.5. The diagnosis report features $\boldsymbol{f}_{diag}$ are concatenated as $n_s \times n_{diag}$ as input features, where n_s is the number of suspects and n_{diag} is the size of diagnosis report features. Firstly, the encoder transfer the origin features to $n_s \times 4$ as latent codes $\boldsymbol{z}$ with five linear layers. The latent codes are used to update the critic-network by minimizing the difference between true rewards and predictions with mean-square loss.

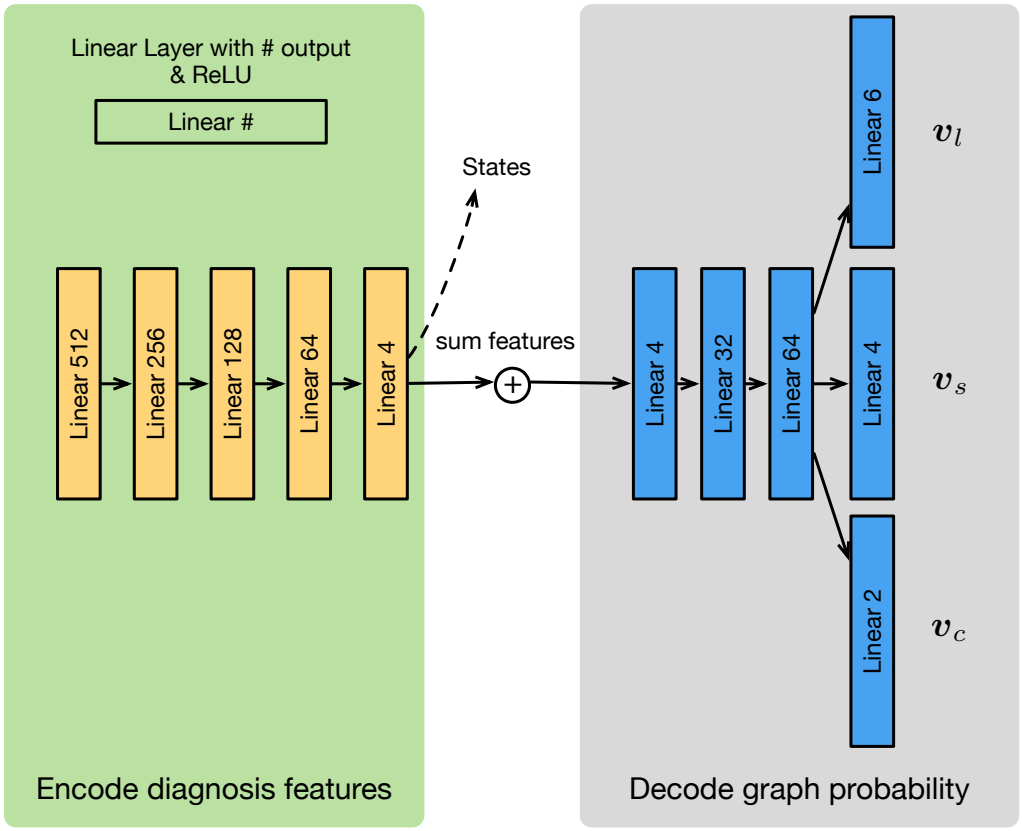


Figure 6.5 Structure of Actor-Network.

Then, features are merged by summation operation as $n_s \times 1$. Finally, features are decoded with three linear layers. $\boldsymbol{v}_s$, $\boldsymbol{v}_c$, and $\boldsymbol{v}_l$ are generated through three branches. Diagnosis report features used in this work are listed in TABLE 6.1. The input features are concatenated as:

$$\left[\ h_j^n \mid s_j^n \mid v_j \mid l \mid t\ \right],$$

where l is the index of layer and t is the label of defect categories. Concatenated features are normalized to improve the numerical stability during training.

Training and Inference. Algorithm 3 describes the training and inference procedures of RCI RL agent. We follow [10] using Deep Layout Snippet Clustering (DLSC) as the feature extractor to generate clusters for a fair comparison. The number of generated clusters k are equal in experimental settings of this work and [10]. Reward functions of A_{score}, A_{cause}, and A_{layer} are used to update the actor during training. During training, the parameters of the actor-network and critic-network are updated through the stochastic optimization method Adam by minimizing the loss functions. At inference stage, the causal graphs g_s, g_c, g_l are generated according the predictions of the actor-network. In line 19, the layout snippets which are not included in g_s, g_c, g_l are removed since they are not the causal reasons to the root causes. Non-relevant layout snippets are filtered out by the judgment of the reinforcement learning agent. In line 25, the optimal root cause distribution is returned according to the

voting results of three distributions.

6.4 Experimental results

The proposed framework is implemented in Python with Pytorch library [65]. We train the framework on one Nvidia Tesla V100 GPU. The implementation of the feature extractor is obtained from the authors of [10]. The effectiveness of our proposed framework is evaluated on six noise-free datasets from six different layout designs, forty noisy datasets from five layout designs, and fifty mixture datasets from five layout designs. Two Adam optimizers [47] are used to train the actor-network and critic-network respectively. At the initial stage of the training, the probability that the root causes are not in the action space is high. We use a cross-entropy loss as the guidance at the first several steps to make sure the robustness of training. We regard the percentage of datasets that the root cause is identified as the accuracy.

Datasets. In this work, the datasets including the diagnosis reports and layout designs presented in [10] are used. The diagnosis reports are generated following the same steps performed in [3]. Defects of type `Open` and `Bridge` are considered in the injection steps. In the evaluation, three classes of datasets are considered:

- **Noise-free dataset.** Case 2 to case 6 are five real silicon datasets that result from real in-production designs. Diag-

Algorithm 3 Training and Inference flow of RCI RL agent.

Input: $R = \{r^n\}_{n=1}^m$ - a set of diagnosis reports, $S = \{\boldsymbol{s}_j\}_{j=1}^n$ - a set of layout snippets of all potential root causes;

function `RL_Agent_Train`(R, S);
  $\boldsymbol{x} \leftarrow$ extract features from R;
  $\boldsymbol{v}_s, \boldsymbol{v}_c, \boldsymbol{v}_l \leftarrow$ Actor$(\boldsymbol{x})$;
  $g_s, g_c, g_l \leftarrow$ generate causal graphs with $\boldsymbol{v}_s, \boldsymbol{v}_c, \boldsymbol{v}_l$;
  Cluster features $\boldsymbol{c} \leftarrow$ DLSC(S);
  Train defect SCM$(\boldsymbol{c})$;
  for $i = 1 \rightarrow k$ **do**
    $\tau_s^i, \tau_c^i, \tau_l^i \leftarrow$ calculate ACE respect to g_s, g_c, g_l; ▷ Equation (6.2)
  end for
  $p_s, p_c, p_l \leftarrow$ get root cause distributions respect to g_s, g_c, g_l; ▷ Equation (6.3)
  $R(g_s, \boldsymbol{r}), R(g_c, \boldsymbol{r}), R(g_l, \boldsymbol{r}) \leftarrow$ get rewards of causal graphs; ▷ Equation (6.9)
  Update Actor and Critic using actor-critic algorithm; ▷ Equation (6.10)
end function
function `RL_Agent_Inference`(R, S);
  $\boldsymbol{x} \leftarrow$ extract features from R;
  $\boldsymbol{v}_s$, $\boldsymbol{v}_c$, $\boldsymbol{v}_l \leftarrow$ Actor$(\boldsymbol{x})$;
  $g_s, g_c, g_l \leftarrow$ generate causal graphs with $\boldsymbol{v}_s, \boldsymbol{v}_c, \boldsymbol{v}_l$;
  $S' \leftarrow$ select layout snippets according to g_s, g_c, g_l;
  Cluster features $\boldsymbol{c}' \leftarrow$ DLSC(S');
  Train defect SCM$(\boldsymbol{c}')$;
  for $i = 1 \rightarrow k$ **do**
    $\tau_s^i, \tau_c^i, \tau_l^i \leftarrow$ calculate ACE respect to g_s, g_c, g_l; ▷ Equation (6.2)
  end for
  $p_* \leftarrow$ get root cause distribution respect to g_*; ▷ Equation (6.3)
  return Root cause distribution p_*;
end function

nosis reports in one noise-free dataset share one single true root cause of the systematic defect. Both open and bridge types are considered in our experiments (TABLE 6.3). The data of each layout design consists of `#Injections` noise-free datasets.

- **Noisy dataset.** In the noisy dataset, a certain percentage of diagnosis reports share a single true root cause and the remaining diagnosis reports have root causes different from the true one (i.e., noise). The sources of noise are sampled from the entire layout designs among all metal layers. Furthermore, we consider the different percentages of noise during the process of injections which are much close to the real cases.

- **Mixture dataset.** Diagnosis reports in this dataset are divided into four portions. Each portion share one root cause independently. Three portions of root causes are true and the rest portion is different from the true one. For example, the proportion '30-30-20-20' in TABLE 6.6 means 30%, 30%, and 20% of diagnosis reports have three true root causes correspondingly and 20% of diagnosis reports have random noise.

Noted that the diagnosis reports of Case 2 are used to train the RCI agent, the diagnosis reports generated by each injection are regarded as one sample.

Table 6.2 Layout Design Information.

	Size($\mu m \times \mu m$)	#Layers
Case 1	8881×9328	5
Case 2	429×384	9
Case 3	8033×7822	6
Case 4	1091×1304	8
Case 5	2300×2410	9
Case 6	1483×1736	7

Table 6.3 Defect Injection Benchmark Statistics.

	#Injections	#Open	#Bridge
Case 1	68	50	18
Case 2	107	72	35
Case 3	93	69	24
Case 4	44	28	16
Case 5	25	18	7
Case 6	39	28	11
Case 2 noise	963	648	315
Case 3 noise	736	544	192
Case 4 noise	221	145	76
Case 5 noise	176	125	51
Case 6 noise	429	356	73

Table 6.4 Accuracy(%) on Noise-free Datasets.

Dataset	Commercial Tool	LPA-DCE [10]	Ours
Case 1	98.53	100.00	100
Case 2	92.52	98.04	100
Case 3	98.92	98.92	100
Case 4	72.09	97.67	97.67
Case 5	82.05	91.3	95.65
Case 6	84.62	94.87	97.44
Average	88.12	96.8	**98.46**
Ratio	0.895	0.983	**1.0**

6.4.1 Comparison with Previous Works.

Experimental results on the noise-free dataset, noisy dataset, and mixture dataset are presented in this section. The Commercial tool and previous work are regarded as the baseline for comparison.

Performance on noise-free dataset. The performance of the RCI RL framework on noise-free datasets is presented in TABLE 6.4. We got 100% accuracy on case 1 to case 3. 4.35% and 2.57% gain on case 5 and case 6 compared to the previous work. The overall performance on six cases is 10.34% higher than the commercial tool and 1.66% better than the result presented in [10].

Performance on noisy dataset. The setting of the noisy dataset is much more challenging than the situation of the noise-free dataset since non-relevant diagnosis reports and layout snip-

pets are mixed into the inputs. Even so, the identification performance of our framework is better than the industry tool and state-of-the-art solution. The proposed framework outperforms the previous work [10] 1.68%, 1.21%, 2.4%, 0.43%, and 0.32% on case 2 to case 6 correspondingly. The industry tool can not identify the root cause in case 5 due to the conflict between the pitch size of layout snippets and tool settings.

Performance on mixture dataset. By evaluating the performance of the proposed framework on mixture datasets, which is a setting much close to the real scenario. The RCI RL framework got a competitive result compared to the previous work and got a much better identification result than the commercial tool. We got 3.63%, 5.5%, 2.7, and 2.26% accuracy gain on case 2, case 3, case 5, and case 6 correspondingly. In case 4, most of the accuracy of different proportion settings are the same between [10] and ours. This is because the root causes are evenly distributed among mixture diagnosis reports and action space is not pruned in most cases.

6.4.2 Runtime

The runtime profiling of the proposed framework is presented in Fig. 6.6. 'RL' indicates the proportion of runtime on reinforcement learning, 'SCM' is the proportion of runtime on the structural causal model. We spend around 10.21% runtime on the RL agent, which is a small portion of additional resource

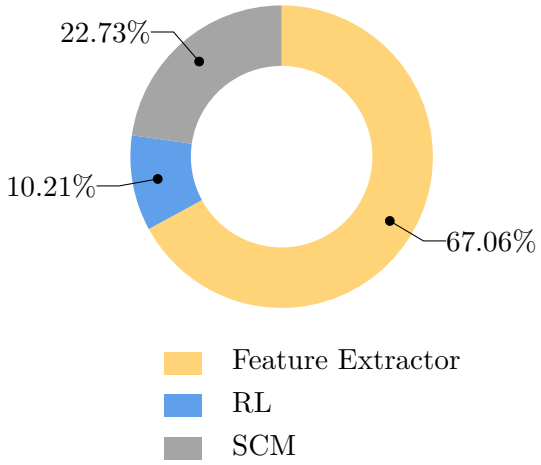


Figure 6.6 Runtime breakdown of our framework.

consumption.

6.4.3 Future Works

The completeness of diagnosis reports is assumed to be equal in all cases during the modeling procedure. A more challenging situation such as designs at the beginning of a production ramp-up stage, where the information is not perfectly available in the diagnosis reports is not discussed in this work. Also, the transferability between diagnosis reports on different designs is worth thinking about. Extending the structural causal models to refer to the diagnosis reports is a challenging topic. We leave these for exploration in future works.

6.5 Discussion and Conclusion

In this paper, we propose a framework to identify the root causes by searching the optimal causal graph. Reinforcement learning with the actor-critic algorithm is used as the guidance of exploitation and exploration scheme. To facilitate the feature learning efficiency, we construct the encoder-decoder network to learn the causal graph. Instead of conventional structural learning predicting the DAG with an adjacency matrix, we consider the critical factors in hierarchical diagnosis reports and formulate a simplified action space. Experimental results on several industry cases show that our framework outperforms the state-of-the-art solutions.

□ **End of chapter.**

Table 6.5 Accuracy(%) on Noisy Datasets.

Noise(%)	Commercial Tool					LPA-DCE [10]					Ours				
	Case 2	Case 3	Case 4	Case 5	Case 6	Case 2	Case 3	Case 4	Case 5	Case 6	Case 2	Case 3	Case 4	Case 5	Case 6
80	70.09	26.88	14.29	-	17.95	94.32	95.7	64	100	86.67	96.59	100	76	100	93.10
70	88.79	44.09	35.71	-	28.21	97.17	92.47	94.59	92	92.31	98.11	96.77	94.59	96	94.87
60	93.46	52.69	50	-	46.15	92.52	97.85	95.12	92	94.87	92.52	97.85	95.12	92	94.87
50	93.46	69.89	52.38	-	64.1	87.85	98.92	88.1	96	89.74	91.59	97.85	95.24	100	89.74
40	93.46	89.25	57.14	-	82.05	87.85	95.7	95.24	100	89.74	89.72	97.85	95.24	95.24	89.74
30	93.46	90.22	61.9	-	74.19	91.59	94.57	100	89.74	96.77	91.59	94.57	97.62	89.74	90.32
20	92.52	90.11	64.29	-	78.57	94.39	98.9	97.62	100	92.86	99.07	98.9	100	94.44	92.86
10	93.46	97.73	66.67	-	84.62	99.07	100	100	94.44	100	99.07	100	100	100	100
Average	89.84	70.11	50.30	-	59.48	93.10	96.76	91.83	95.49	92.87	**94.78**	**97.97**	**94.23**	**95.92**	**93.19**
Ratio	0.948	0.716	0.534	-	0.638	0.982	0.988	0.975	0.996	0.997	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Table 6.6 Accuracy(%) on Mixture Datasets.

Proportion (r1%-r2%-r3%-noise%)	Commercial Tool					LPA-DCE [10]					Ours				
	Case 2	Case 3	Case 4	Case5	Case6	Case 2	Case 3	Case 4	Case5	Case6	Case 2	Case 3	Case 4	Case5	Case6
20-20-20-40	62.07	8.05	29.21	-	13.83	77.19	78.02	64	79.37	86.32	85.96	93.41	64	74.6	87.37
30-20-20-30	65.91	19.28	29.03	-	19.57	82.35	89.01	82.61	72.6	83.15	82.35	92.31	82.61	79.45	83.15
30-30-20-20	71.74	38.82	32.22	-	41.94	80.23	86.81	78.26	84.34	91.11	93.02	92.31	78.26	84.34	91.11
30-30-30-10	79.17	74.42	40	-	45.83	89.47	93.41	92.31	87.88	87	86.32	96.7	92.31	87.88	91
40-20-20-20	70.79	27.06	30	-	38.71	84.51	85.71	69.23	81.43	91.11	84.51	91.21	69.23	81.43	91.11
40-30-20-10	69.15	62.79	35.23	-	45.45	82.76	83.52	84.09	92.05	91.3	88.51	98.9	84.09	95.45	91.3
40-30-30-0	83.51	81.32	42.42	-	55	91.84	96.7	89.19	89	87	92.85	96.7	86.49	89	95
50-20-20-10	70.65	61.63	31.33	-	32.63	72.46	90.11	78.79	70.27	89.16	81.16	90.11	78.79	70.27	91.57
50-30-20-0	73.12	84.27	40.22	-	46	83.13	91.21	90.24	83.7	95.45	85.54	91.21	90.24	92.39	95.45
60-20-20-0	72.22	74.71	37.08	-	37.11	81.54	82.42	83.78	68.57	86.9	81.54	89.01	83.78	81.43	94.05
Average	71.83	53.24	34.67	-	37.61	82.55	87.69	**81.25**	80.92	88.85	**86.18**	**93.19**	80.98	**83.62**	**91.11**
Ratio	0.833	0.571	0.428	-	41.28	0.958	0.941	**1.003**	0.968	0.975	**1.0**	**1.0**	1.0	**1.0**	**1.0**

Chapter 7

Conclusion

In the thesis, several Artificial Intelligence methodologies are introduced to solve layout defect problems in design for manufacturability. We discuss the lithography defect detection at the post-layout stage and take a step further to identify the systematic defects. In this chapter, we first conclude each proposed methodology and then discuss the future extensions.

7.1 summary

The reduction of quality in circuits is mainly attributed to variations in lithography. As an alternative to resolution enhancement technologies, layout hotspot detection can also ease the variations. Existing hotspot detectors only work on small clips extracted from a whole chip layout and can only detect one hotspot location at a time that occurs at the center of each clip. A conventional hotspot detection scheme requires repeatedly scanning overlapping regions of a full chip design. In Chap-

ter 4, we propose a new faster region-based hotspot detection framework to avoid the waste of computational resources and time-consuming when facing extremely large layouts. The proposed framework can mark multiple hotspot locations within a region that is much larger than a clip applied in previous work. A regression and classification multi-task flow is implemented in the framework to guide to higher accuracy, higher detection speed, and lower false alarm penalties. To further improve accuracy and reduce false alarms, a clip proposal network and refinement stage are built. We introduce a multi-branch design for encoder-decoder and IoU regularization to further enhance the proposed region-based hotspot detector.

Detecting hotspots across entire layouts may result in over-correction, which negatively affects chip area and performance. Traditional methods, such as Physical failure analysis (PFA), are able to reveal whether a layout pattern is the root cause of systematic defects. While it is typically time-consuming and costly and requires both experience and knowledge of fabrication processes. In Chapter 5, a unified framework for layout pattern analysis with deep causal effect estimation is proposed. Our framework characterizes the causal relationship between potential root causes and the systematic defect. By using contrastive learning we train an encoder to extract from layout snippets rotation-, mirror-, and shift-invariant latent features. In experiments on large-scale designs, our framework achieves state-of-the-art results which significantly outperform commercial tools

in terms of accuracy and inference time. We propose a reinforcement learning-based framework to improve the root cause identification quality in Chapter 6 to solve the drawback of empirical causal graphs. The principle of reinforcement learning allows it to select problem-specific combinations of search strategies and scores, which is an advantage to search for the optimal causal graph.

7.2 Possible Future Directions

Concerning hotspot detection, more advanced machine learning and deep learning techniques will be introduced and customized to aid the detection flow. Layout features under different technology nodes should be considered during modeling the learning problems. Besides, multi-layer layout hotspot detection and full-chip scale detection are promising directions, and layout clippers can be further investigated.

For the techniques in layout pattern analysis, as mentioned in the experimental result part of Chapter 5, more exploration at the gate level is required since identifying the root cause at an earlier stage has great benefits to the yields. Layout pattern analysis can identify the layout level defects such as open and bridge. Adding layout information to the diagnosis helps to improve failure analysis by reducing turnaround time and cost. But, as the industry moves into the FinFET era, product engineers are finding that the established scan test diagnosis needs

an upgrade. We see more front-end-of-line (FEOL) defects at the transistor level rather than in the interconnect, and fin-related defects tend to be timing-related, making physical failure analysis and yield analysis far more difficult challenges than in the past. Also, considering the probability of low fidelity on diagnosis reports, identifying the root causes at the yield ramp-up stage is an interesting direction we need to explore.

Apart from the above, limited by the number of datasets in academia, methodologies on data generation and argumentation are necessary. Augmentation strategies, layout generation methods, and related methods also need to concern. More advanced methodologies will be developed in the future.

□ **End of chapter.**

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al. TensorFlow: A system for large-scale machine learning. In *Proc. OSDI*, pages 265–283, 2016.

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] B. Benware, C. Schuermyer, M. Sharma, and T. Herrmann. Determining a failure root cause distribution from a population of layout-aware scan diagnosis results. *IEEE Design & Test of Computers*, 29(1):8–18, 2012.

[4] Z. Cai and N. Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *Proc. CVPR*, pages 6154–6162, 2018.

[5] A. Chattopadhyay, P. Manupriya, A. Sarkar, and V. N. Balasubramanian. Neural network attributions: A causal perspective. In *International Conference on Machine Learning*, pages 981–990. PMLR, 2019.

[6] J. Chen, Y. Lin, Y. Guo, M. Zhang, M. B. Alawieh, and D. Z. Pan. Lithography hotspot detection using a double inception module architecture. *JM3*, 18(1):013507, 2019.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[9] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. ECCV*, pages 801–818, 2018.

[10] R. Chen, S. Hu, Z. Chen, S. Zhu, B. Yu, P. Li, C. Chen, Y. Huang, and J. Hao. A unified framework for layout pattern analysis with deep causal estimation. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2021.

[11] R. Chen, W. Zhong, H. Yang, H. Geng, F. Yang, X. Zeng, and B. Yu. Faster region-based hotspot detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[12] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu. Faster region-based hotspot detection. In *Proc. DAC*, pages 1–6, 2019.

[13] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[14] X. Chen and K. He. Exploring simple siamese representation learning, 2020.

[15] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan. Semi-supervised hotspot detection with self-paced multi-task learning. In *Proc. ASPDAC*, pages 420–425, 2019.

[16] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan. Semi-supervised hotspot detection with self-paced multi-task learning. *IEEE TCAD*, 2019.

[17] W.-T. Cheng, R. Klingenberg, B. Benware, W. Yang, M. Sharma, G. Eide, Y. Tian, S. M. Reddy, Y. Pan, S. Fernandes, et al. Automatic identification of yield limiting layout patterns using root cause deconvolution on volume scan diagnosis data. In *2017 IEEE 26th Asian Test Symposium (ATS)*, pages 219–224. IEEE, 2017.

[18] W.-T. Cheng, Y. Tian, and S. M. Reddy. Volume diagnosis data mining. In *2017 22nd IEEE European Test Symposium (ETS)*, pages 1–10. IEEE, 2017.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li. Imagenet: A large-scale hierarchical image database. In *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[20] D. Ding, J. A. Torres, and D. Z. Pan. High performance lithography hotspot detection with successively refined pattern identifications and machine learning. *IEEE TCAD*, 30(11):1621–1634, 2011.

[21] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan. EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation. In *Proc. ASPDAC*, pages 263–270, 2012.

[22] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1422–1430, 2015.

[23] D. G. Drmanac, F. Liu, and L.-C. Wang. Predicting variability in nanoscale lithography processes. In *Proc. DAC*, pages 545–550, 2009.

[24] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. CenterNet: Object detection with keypoint triplets. In *Proc. ICCV*, pages 6569–6578, 2019.

[25] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[26] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. He. A large-scale study on unsupervised spatiotemporal representation learning. *arXiv preprint arXiv:2104.14558*, 2021.

[27] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[28] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[29] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

[30] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proc. CVPR*, pages 3354–3361, 2012.

[31] D. Geiger and D. Heckerman. Learning gaussian networks. In *Uncertainty Proceedings 1994*, pages 235–243. Elsevier, 1994.

[32] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng. Sparse VLSI layout feature extraction: A dictionary learning approach. In *Proc. ISVLSI*, pages 488–493, 2018.

[33] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu. Hotspot detection via attention-based deep layout metric learning. In *Proc. ICCAD*, 2020.

[34] R. Girshick. Fast R-CNN. In *Proc. ICCV*, pages 1440–1448, 2015.

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, pages 580–587, 2014.

[36] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[38] Z. He, Y. Ma, L. Zhang, P. Liao, N. Wong, B. Yu, and M. D. Wong. Learn to floorplan through acquisition of effective local search heuristics. In *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pages 324–331. IEEE, 2020.

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[40] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proc. CVPR*, pages 7310–7311, 2017.

[41] L. Huang, Y. Yang, Y. Deng, and Y. Yu. DenseBox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[42] X. Huang, M. Qin, R. Xu, C. Chen, S. Jui, Z. Ding, P. Li, and Y. Huang. Adaptive nn-based root cause analysis in volume diagnosis for yield improvement. In *2021 IEEE International Test Conference (ITC)*, pages 30–36. IEEE, 2021.

[43] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng. Efficient layout hotspot detection via binarized residual neural network. In *Proc. DAC*, page 147, 2019.

[44] Z. Jie, X. Liang, J. Feng, W. F. Lu, E. H. F. Tay, and S. Yan. Scale-aware pixelwise object proposal networks. *IEEE Transactions on Image Processing (TIP)*, 25(10):4525–4539, 2016.

[45] A. B. Kahng, C.-H. Park, and X. Xu. Fast dual graph based hotspot detection. In *Proc. SPIE*, volume 6349, 2006.

[46] J. Kim and M. Fan. Hotspot detection on Post-OPC layout using full chip simulation based verification tool: A case study with aerial image simulation. In *Proc. SPIE*, volume 5256, 2003.

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

[48] T. Kong, F. Sun, H. Liu, Y. Jiang, and J. Shi. FoveaBox: Beyond anchor-based object detector. *arXiv preprint arXiv:1904.03797*, 2019.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. NIPS*, pages 1097–1105, 2012.

[50] H. Law and J. Deng. CornerNet: Detecting objects as paired keypoints. In *Proc. ECCV*, pages 734–750, 2018.

[51] H. Liao, W. Zhang, X. Dong, B. Poczos, K. Shimada, and L. Burak Kara. A deep reinforcement learning approach for global routing. *Journal of Mechanical Design*, 142(6), 2020.

[52] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, pages 2117–2125, 2017.

[53] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proc. CVPR*, pages 2117–2125, 2017.

[54] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proc. ICCV*, pages 2980–2988, 2017.

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014.

[56] Y. Lin, T. Qu, Z. Lu, Y. Su, and Y. Wei. Asynchronous reinforcement learning framework and knowledge transfer for net order exploration in detailed routing. *IEEE Trans-*

actions on Computer-Aided Design of Integrated Circuits and Systems, 2021.

[57] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proc. ECCV*, pages 21–37, 2016.

[58] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan. A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In *Proc. SPIE*, volume 9427, 2015.

[59] T. Matsunawa, S. Nojima, and T. Kotani. Automatic layout feature extraction for lithography hotspot detection based on deep neural network. In *SPIE Advanced Lithography*, volume 9781, 2016.

[60] T. Matsunawa, B. Yu, and D. Z. Pan. Laplacian eigenmaps-and bayesian clustering-based layout pattern sampling and its applications to hotspot detection and optical proximity correction. *JM3*, 15(4):043504–043504, 2016.

[61] A. Mirhoseini, A. Goldie, M. Yazgan, J. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, S. Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[62] I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6707–6717, 2020.

[63] S. S. Mousavi, M. Schukat, and E. Howley. Deep reinforcement learning: an overview. In *Proceedings of SAI Intelligent Systems Conference*, pages 426–440. Springer, 2016.

[64] D. Z. Pan, B. Yu, and J.-R. Gao. Design for manufacturing with emerging nanolithography. *IEEE TCAD*, 32(10):1453–1472, 2013.

[65] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proc. NIPS*, pages 8024–8035, 2019.

[66] D. Patel, R. Bonam, and A. A. Oberai. Engineering neural networks for improved defect detection and classification. In *Proc. SPIE*, volume 10959, 2019.

[67] J. Pearl. *Causality*. Cambridge university press, 2009.

[68] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proc. CVPR*, pages 779–788, 2016.

[69] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Proc. CVPR*, pages 7263–7271, 2017.

[70] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[71] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. NIPS*, pages 91–99, 2015.

[72] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):1137–1149, 2017.

[73] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union. June 2019.

[74] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proc. CVPR*, pages 658–666, 2019.

[75] E. Roseboom, M. Rossman, F.-C. Chang, and P. Hurat. Automated full-chip hotspot detection and removal flow

for interconnect layers of cell-based designs. In *Proc. SPIE*, volume 6521, 2007.

[76] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[77] M. Shin and J.-H. Lee. Accurate lithography hotspot detection using deep convolutional neural networks. *JM3*, 15(4):043507, 2016.

[78] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pages 3145–3153. PMLR, 2017.

[79] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.

[80] D. Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.

[81] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR, 2017.

[82] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual

connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[83] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, pages 2818–2826, 2016.

[84] H. Takahashi, H. Ogura, S. Sato, A. Takahashi, and C. Kodama. A feature selection method for weak classifier based hotspot detection. In *Proc. SPIE*, volume 11328, 2020.

[85] W. C. Tam, O. Poku, and R. D. Blanton. Systematic defect identification through layout snippet clustering. In *2010 IEEE International Test Conference*, pages 1–10. IEEE, 2010.

[86] W. C. J. Tam and R. D. S. Blanton. Lasic: Layout analysis for systematic ic-defect identification using clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1278–1290, 2015.

[87] H. Tang, S. Manish, J. Rajski, M. Keim, and B. Benware. Analyzing volume diagnosis results with statistical learning for yield improvement. In *12th IEEE European Test Symposium (ETS'07)*, pages 145–150. IEEE, 2007.

[88] Z. Tian, C. Shen, H. Chen, and T. He. FCOS: Fully convolutional one-stage object detection. In *Proc. ICCV*, pages 9627–9636, 2019.

[89] Y. Tomioka, T. Matsunawa, C. Kodama, and S. Nojima. Lithography hotspot detection by two-stage cascade classifier using histogram of oriented light propagation. In *Proc. ASPDAC*, pages 81–86, 2017.

[90] R. O. Topaloglu. ICCAD-2016 CAD contest in pattern classification for integrated circuit design space analysis and benchmark suite. In *Proc. ICCAD*, pages 41:1–41:4, 2016.

[91] A. J. Torres. ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *Proc. ICCAD*, pages 349–350, 2012.

[92] S. S.-E. Tseng, W.-C. Chang, I. H.-R. Jiang, J. Zhu, and J. P. Shiely. Efficient search of layout hotspot patterns for matching sem images using multilevel pixelation. In *Proc. SPIE*, volume 10961, page 109610B, 2019.

[93] D. P. Vassileios Balntas, Edgar Riba and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In E. R. H. Richard C. Wilson and W. A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 119.1–119.11. BMVA Press, September 2016.

[94] S. I. Vrieze. Model selection and psychological theory: a discussion of the differences between the akaike informa-

tion criterion (aic) and the bayesian information criterion (bic). *Psychological methods*, 17(2):228, 2012.

[95] J. Wang, K. Chen, S. Yang, C. C. Loy, and D. Lin. Region proposal by guided anchoring. In *Proc. CVPR*, pages 2965–2974, 2019.

[96] X. Wang, Y. Du, S. Zhu, L. Ke, Z. Chen, J. Hao, and J. Wang. Ordering-based causal discovery with reinforcement learning. *arXiv preprint arXiv:2105.06631*, 2021.

[97] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE TCAD*, 33(11):1671–1680, 2014.

[98] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[99] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *Proc. DAC*, pages 131:1–131:6, 2018.

[100] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu. Imbalance aware lithography hotspot detection: a deep learning approach. *JM3*, 16(3):033504, 2017.

[101] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu. Detecting multi-layer layout hotspots with adaptive squish patterns. In *Proc. ASPDAC*, pages 299–304, 2019.

[102] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE TCAD*, 38(6):1175–1187, 2019.

[103] M. Yang, F. Liu, Z. Chen, X. Shen, J. Hao, and J. Wang. Causalvae: Disentangled representation learning via neural structural causal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9593–9602, 2021.

[104] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin. RepPoints: Point set representation for object detection. In *Proc. ICCV*, pages 9657–9666, 2019.

[105] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai. Efficient process-hotspot detection using range pattern matching. In *Proc. ICCAD*, pages 625–632, 2006.

[106] W. Ye, M. B. Alawieh, M. Li, Y. Lin, and D. Z. Pan. Lithogpa: Gaussian process assurance for lithography hotspot detection. In *Proc. DATE*, pages 54–59, 2019.

[107] W. Ye, M. B. Alawieh, M. Li, Y. Lin, and D. Z. Pan. LithoGPA: Gaussian process assurance for lithography hotspot detection. In *Proc. DATE*, pages 54–59, 2019.

[108] W. Ye, Y. Lin, M. Li, Q. Liu, and D. Z. Pan. LithoROC: lithography hotspot detection with explicit ROC optimization. In *Proc. ASPDAC*, pages 292–298, 2019.

[109] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan. Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering. *JM3*, 14(1):011003, 2015.

[110] B. Yu, X. Xu, S. Roy, Y. Lin, J. Ou, and D. Z. Pan. Design for manufacturability and reliability in extreme-scaling VLSI. *Science China Information Sciences*, pages 1–23, 2016.

[111] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia (MM)*, pages 516–520, 2016.

[112] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 516–520, 2016.

[113] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang. Accurate process-hotspot detection using critical design rule extraction. In *Proc. DAC*, pages 1167–1172, 2012.

[114] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang. Machine-learning-based hotspot detection using topological classification and critical feature extraction. *IEEE TCAD*, 34(3):460–470, 2015.

[115] H. Zhang, B. Yu, and E. F. Y. Young. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. In *Proc. ICCAD*, pages 47:1–47:8, 2016.

[116] H. Zhang, F. Zhu, H. Li, E. F. Y. Young, and B. Yu. Bilinear lithography hotspot detection. In *Proc. ISPD*, pages 7–14, 2017.

[117] X. Zhou, J. Zhuo, and P. Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proc. CVPR*, pages 850–859, 2019.

[118] C. Zhu, Y. He, and M. Savvides. Feature selective anchor-free module for single-shot object detection. In *Proc. CVPR*, 2019.

[119] S. Zhu, I. Ng, and Z. Chen. Causal discovery with reinforcement learning. In *International Conference on Learning Representations*, 2019.