Ph.D. Dissertation of Engineering

A novel deep architecture for image segmentation in photolithography inspection systems

포토리소그래피 검사 시스템의 이미지 분할을 위한 새로운 깊은 아키텍처

August 2021

Graduate School of
Convergence Science and Technology
Seoul National University
Program in Intelligent Systems

Junghee Han

A novel deep architecture for image segmentation in photolithography inspection systems

Advisor Seongsoo Hong

Submitting a Ph.D. Dissertation of Engineering

August 2021

Graduate School of
Convergence Science and Technology
Seoul National University
Program in Intelligent Systems

Junghee Han

Confirming the Ph.D. Dissertation written by
Junghee Han
August 2021

Chair	박 재 흥
Vice Chair	홍 성 수
Examiner	곽 노 준
Examiner	김 세 화
Examiner	전 동 석

Abstract

In semiconductor manufacturing, defect detection is critical to maintain high yield. Typically, the defects of semiconductor wafer may be generated from the manufacturing process. Most computer vision systems used in semiconductor photolithography process inspection still have adopt to image processing algorithm, which often occur inspection faults due to sensitivity to external environment changes. Therefore, we intend to tackle this problem by means of converging the advantages of image processing algorithm and deep learning.

In this dissertation, we propose Image Segmentation Detector (ISD) to extract the enhanced feature-maps under the situations where training dataset is limited in the specific industry domain, such as semiconductor photolithography inspection. ISD is used as a novel backbone network of state-of-the-art Mask R-CNN framework for image segmentation. ISD consists of four dense blocks and four transition layers. Especially, each dense block in ISD has the shortcut connection and the concatenation of the feature-maps produced in layer with dynamic growth rate for more compactness. ISD is trained from scratch without using recently approached transfer learning method. Additionally, ISD is trained with image dataset pre-

processed by means of our designed image filter to extract the better enhanced feature map of Convolutional Neural Network (CNN). In ISD, one of the key design principles is the compactness, plays a critical role for addressing real-time problem and for application on resource bounded devices.

To empirically demonstrate the model, this dissertation uses the existing image obtained from the computer vision system embedded in the currently operating semiconductor manufacturing equipment. ISD achieves consistently better results than state-of-the-art methods at the standard mean average precision which is the most common metric used to measure the accuracy of the instance detection. Significantly, our ISD outperforms baseline method DenseNet, while requiring only 1/4 parameters. We also observe that ISD can achieve comparable better results in performance than ResNet, with only much smaller 1/268 parameters, using no extra data or pre-trained models. Our experimental results show that ISD can be useful to many future image segmentation research efforts in diverse fields of semiconductor industry which is requiring real-time and good performance with only limited training dataset.

Keyword : Photolithography inspection, backbone network, image segmentation, deep learning, convolutional neural networks, computer vision.

Student Number : 2010-22695

Contents

List of Figures

List of Tables

Chapter 1. Introduction

Semiconductor manufacturing has emerged as one of the most important world industries. In semiconductor manufacturing, defect detection is critical to maintain high yield. Even with the highly automated and precisely monitored facilities used to process the complex manufacturing steps in a near particle free environment, wafer defect still exist. The causes of defect may be generated from equipment malfunctions in delicate and difficult processing steps. In order to be competitive in the semiconductor manufacturing industry, the detection of these problems becomes a critical issue. In this dissertation, In order to identify and tackle the problems of computer vision systems used to inspect the malfunction of equipment in the semiconductor photolithography process, the-state-of-art technologies are researched, converged and applied.

Object detection or localization is one of the most important and challenging branches of computer vision, which has been widely

applied in people’ s life, such as monitoring security, autonomous driving and military field, transportation field, medical field, industry field so on, with the purpose of locating instances of semantic objects of a certain class. Detection or object localization is an incremental step from coarse to fine inference, which provides not only classes of the image objects but also gives the location of the classified image objects in the form of bounding boxes or centroids.

Semantic segmentation [1, 2] gives fine inference by predicting labels for every pixel in the input image. Each pixel is labelled according to the object class within which it is enclosed. Furthering this evolution, instance segmentation gives different labels for separate instances of objects belonging to the same class. Hence, instance segmentation may be defined as the technique of simultaneously solving the problem of object detection as well as that of semantic segmentation.

Instance segmentation methods that focus on detection bounding box proposals, ignore the classes that are not well suited for detection, e.g., background, scenery. On the other hand, semantic segmentation does not provide instance boundaries for classes in a given image. Furthermore, panoptic segmentation task, first coined by Kirillov et al. [3] unifies these tasks and defines an ideal output for thing classes as instance segmentations, as well as for stuff classes as semantic segmentation.

These segmentation are developing a technique/algorithm that performs well in the two domains of better segmentation accuracy

and better segmentation efficiency. Better segmentation accuracy encompasses accurate localization and recognition of objects in images/frames, with the result that the large variety of object related categories in real scenario can be distinguished (i.e. better distinctiveness), and that instances of objects belonging to same class which are subject to intra-class appearance variation, may be localized and recognized (i.e. better robustness). Better segmentation efficiency refers to computational cost of the segmentation algorithm. It refers efficient real-time computational costs like acceptable memory/storage requirements, and lesser burden on the processor(s). One of the important components in an object detector for segmentation is good feature representation which is of primary importance in object detection. Methods based on deep learning [4] (like Deep CNNs) are able to learn powerful representations of features with various abstraction levels from images. Subsequently, the problem of feature representation has been transferred to the development of better performing network architectures and more optimized training procedures. Deep CNN based detectors like RCNN [5], Fast RCNN [6], Faster RCNN [7], Mask R-CNN [8] and YOLO [9], usually use the deep CNN architectures and subsequently use features from the topmost CNN layer for object representation. However, there are many issues e.g., various scale, geometric transformations, occlusions, image degradations to be addressed. Therefore, in order to good feature representation, the objective of this dissertation is to present a novel

deep architecture that convergences image processing technology and deep learning image segmentation technology, which is strong against image degradation among these problems and can be applied with better efficiency to semiconductor photolithography inspection systems.

As mentioned earlier, we introduced object detection, localization, segmentation, issues of segmentation and the objective to be covered in this dissertation.

1.1. Background and Motivation

Instance segmentation has come to be one of the relatively important, complex and challenging areas in machine vision research. Aimed at predicting the object class-label and the pixel-specific object instance-mask, it localizes different classes of object instances present in various images. Instance segmentation aims to help largely robotics, autonomous driving, surveillance etc. Specifically, instance segmentation is an important task for biomedical and biological image analysis. Due to the complicated background components, the high variability of object appearances, numerous overlapping objects, and ambiguous object boundaries, this task still remains challenging. Recently, Jiao et al [10] list the traditional and new applications of deep learning based object detection (instance segmentation).

However, there is no application in semiconductor industry due to specific process domain.

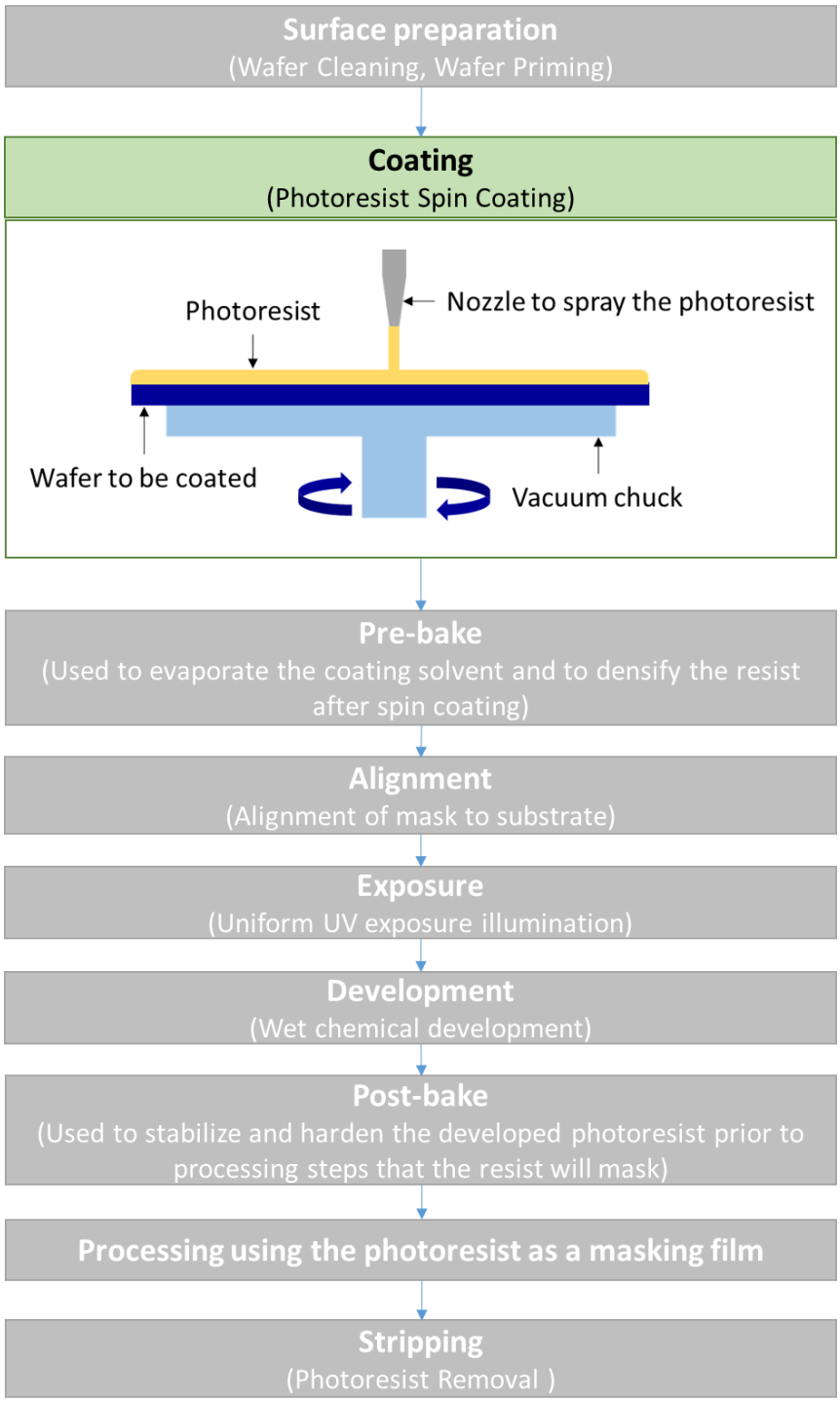


Figure 1.1 Semiconductor photolithography process.

As illustrated in Figure 1.1, the semiconductor photolithography is a process of drawing semiconductor circuits on wafers, coating them thinly with photosensitive polymer materials that respond to light on wafers, then placing a mask on top of the desired pattern and pecking the light to form the desired pattern.

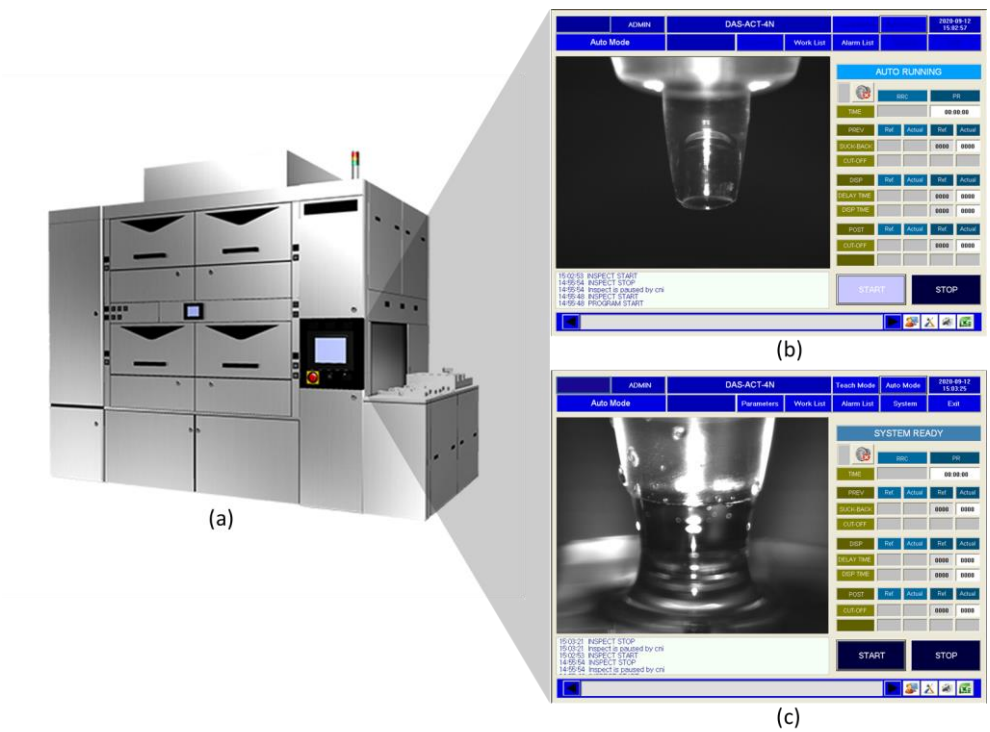
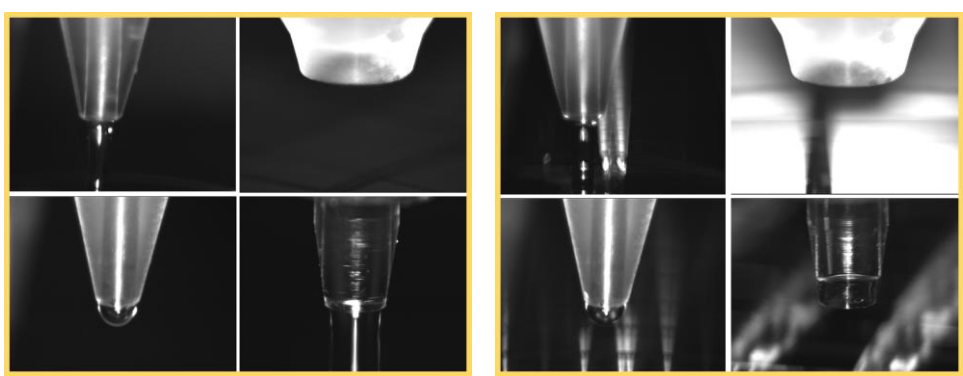


Figure 1.2 The computer vision system embedded in the currently operating semiconductor manufacturing equipment for photolithography inspection. (a) The semiconductor equipment with embedded computer vision system, providing sophisticated process control and techniques in the photomask manufacturing process. (b) An example of suck-back state monitoring in computer vision system. (c) An example of dispensing state monitoring in computer vision system.

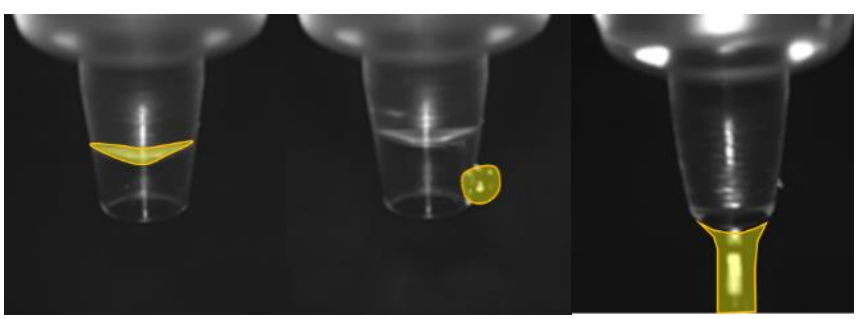
In this process, the photoresist spin coating is used to spread the required thickness of the photoresist uniformly on the wafer. Therefore, the spin coating is an important process. If inspection faults occur in this process, a defective product is produced no matter how well the subsequent process is performed. It is greatly affecting the defect rate in wafer-based process. Thus, in this study, we focus on photoresist spin coating process of semiconductor photolithography. As illustrated in Figure 1.2, the computer vision system is used to prevent defects in semiconductor products by monitoring these processes and predicting defects in the photo process in advance. Generally, the computer vision system uses the digital image processing [11-20] to try and perform emulation of vision at human scale. The computer vision system used in the process of spin coating also finds defects through digital image processing algorithm.



(a) (b)

Figure 1.3 An example of image distorted by external environment factors: (a) Normal image; (b) Distorted image.

However, many detection errors occur due to external environmental factors such as various types of wafers and photoresist, motor rotation speed, and diffuse reflection of light. Figure 1.3 illustrates an example of image distorted by external environment factors. Digital image processing algorithm has high performance in case of images with little influence on the external environment. However, performance is extremely degraded when image distortion occurs due to the external environment. Therefore, in the computer vision system, if the characteristics of the image is changed or distorted, there is a disadvantage in that a new or modified technique of digital image processing algorithm and the specialized signal processing method should be applied to overcome it. To overcome the influence of various image distortion, we adopt deep learning image segmentation technology that is robust even in the external environment.



(a) (b) (c)

Figure 1.4 Three inspection type for detecting defects in the spin coating process of semiconductor photolithography: (a) Suck-back state; (b) Contamination state; (c) Dispensing state.

As illustrated in Figure 1.4, there are three inspection type for detecting defects in the spin coating process of semiconductor photolithography: First is the suck-back state of the nozzle that sprays the photoresist, Second is the contamination state of the nozzle, and Third is the dispensing state to measure the time to spray the photoresist. However, among these, we do not include inspecting contamination state shown in Figure 1.4 (b) in this study, because sufficient images cannot be obtained to perform this study. We leave it for future work. In this dissertation, we propose a method for detecting defects by monitoring the suck-back state and the dispensing state. Therefore, in order to this, it is necessary to find a specific area in an image and extract features within the area to determine whether the defect is defective. Deep learning techniques [21] that can detect specific areas in an image have object detection, semantic segmentation, and instance segmentation. Among them, the instance segmentation technique can be applied to image segmentation for inspecting not only the suck-back state but also the dispensing state.

Image segmentation is a computer vision process designed to simplify image analysis by splitting input into segments that represent objects or parts of objects and form a collection of pixels. Instance segmentation is a subtype of image segmentation which identifies each instance of each object within the image at the pixel level. Instance segmentation can also be thought as object detection where the output is a mask instead of just a bounding box. Agarwal

et al. [22] presented recent advances in object detection in the age of deep convolutional neural networks. The objective of instance segmentation is to detect specific objects in an image and create a mask around the object of interest. Thus, in this work, instance segmentation is used to image segmentation required for semiconductor photolithography inspection.

In computer vision, transfer learning is usually expressed through the use of pre-trained models. To achieve desired performance, the common practice in advanced instance segmentation systems is to fine-tune models pre-trained on ImageNet [23]. This fine-tuning process can be viewed as transfer learning [24-29]. Researchers usually train CNN models on large scale classification datasets like ImageNet [23] first, then fine-tune the models on target tasks, such as object detection [5-7, 9, 30-40], image segmentation [41-44], etc. However, we directly train model without involving any other additional data or extra fine-tuning process. There are numerous state-of-the-art pre-trained CNN models available. Fine-tuning on pre-trained models can quickly convergence to a final state and requires less instance-level annotated training data than basic classification task. As is well-known, fine-tuning can mitigate the gap between different target category distributions. However, it is still a severe problem when the source domain (e.g., ImageNet) has a huge mismatch to the target domain such as industrial images, medical images, etc. As illustrated in Figure 1.4, the image used for inspection is completely different

from the image on source domain (e.g., ImageNet). Without having enough number of dataset, deep artificial neural networks cannot be trained well and it is difficult to collect enough image data in the specific industry domain.

In this work, we investigate three questions. First, is it possible to train image segmentation networks from scratch directly with only smaller dataset without the pre-trained models? Second, are there any principles to design a resource efficient network structure for image segmentation, meanwhile keeping high detection accuracy? Third, is there any methodology to improve inspection performance other than network design? To meet this goal, we propose image segmentation detector (ISD) and pre-processing that is performed by using image filter before training.

Chapter 2. Related Work

In this section, we first summarize current inspection method of computer vision system used in currently operating semiconductor photolithography process for defect detection. We also review the state-of-the-art deep learning based instance segmentation and backbone structure. Then, we review the development of improved network structure for extracting enhanced feature map. Finally, we briefly introduce the metrics adopted to assess the detections in most competitions, and the method for learning from scratch.

2.1. Inspection Method

Efficacious recognition and consistent identification of visual features is an important problem in applications, such as Pattern Recognition,

Structure from motion, Image Registration and Visual Localization. The input data takings, numerous arrangements such as audiovisual arrangements, interpretations from manifold cameras or multidimensional statistics from a scanner. Concurrent performance is a perilous demand to utmost of these applications, which necessitate the finding and corresponding of the visual features in real-time. Although feature recognition and empathy approaches have been deliberate in the work due to their computational intricacy therefore pure software execution by unique hardware is far suitable in their performance for real-time applications. As illustrated in Figure 2.1, Computer vision is related to various other fields, however it is closely linked to the field of image processing. Computer vision systems [45-51] are widely used for on-line inspection and quality control to improve the finished product quality and lower the costs in various industries.

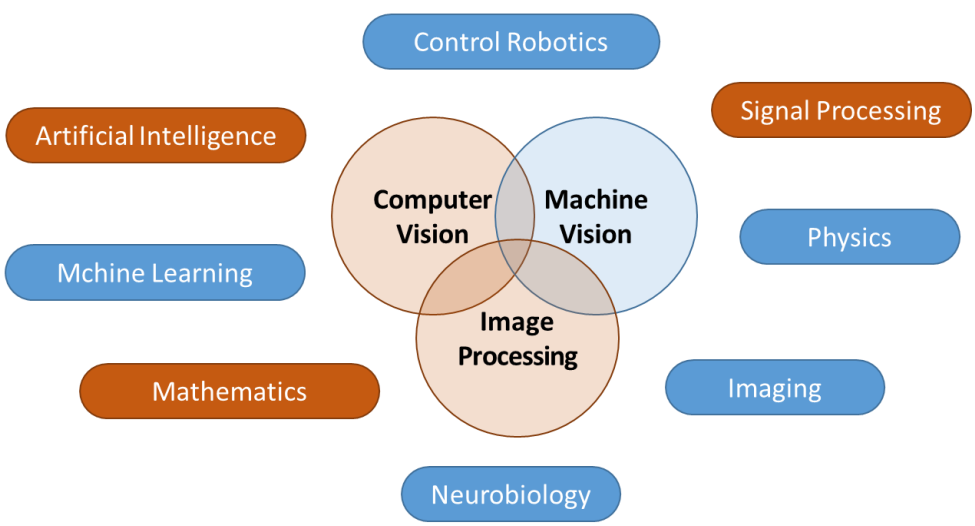


Figure 2.1 Relationship between Computer Vision and Various Other Fields.

The computer vision system used in existing semiconductor photolithography process, performs the specialized digital image processing and signal processing to extract features necessary for defect detection, and determines the defect by means of a neural network as a classifier. The specialized digital image processing removes noise from the input image of specific domain, improves brightness or contrast, emphasizes edges, and makes the image more clearly to extract features. Feature extraction is obtained by the signal processing method that calculates the sum of the vertical component pixels and the horizontal components of the pre-processed image by means of digital image processing, and applies an adaptive threshold. Recognizing the extracted features and determining whether there are defects is composed of a neural network. Figure 2.2 (c) illustrates an example of automatically detecting the contamination state of nozzle by means of digital image processing.

(a) (b) (c)

Figure 2.2 An example of detecting the contamination state of nozzle by means of the specialized digital image processing: (a) Original image; (b) Pre-processed Image; (c) Image where contamination is detected.

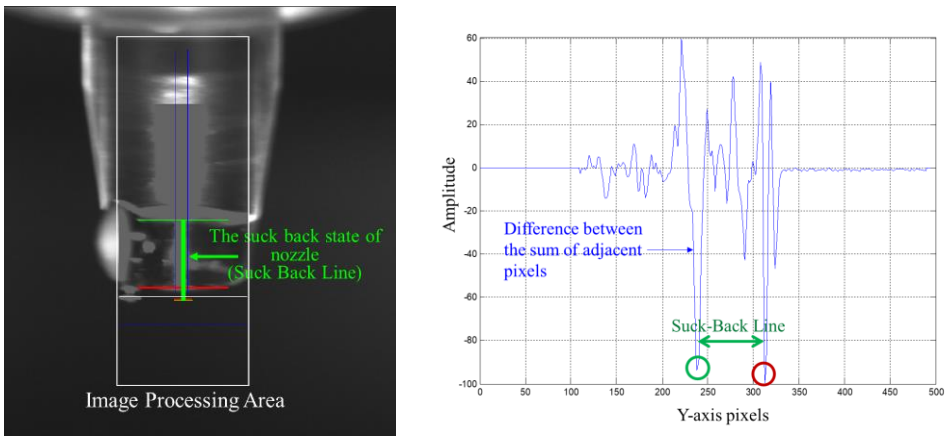


Figure 2.3 An example of detecting the suck-back state of nozzle by means of the specialized signal processing: (a) The suck-back line is detected by means of filtering image within processing area; (b) The suck-back line is detected by means of signal processing which is adopting adaptive threshold and sum of pixels in x direction.

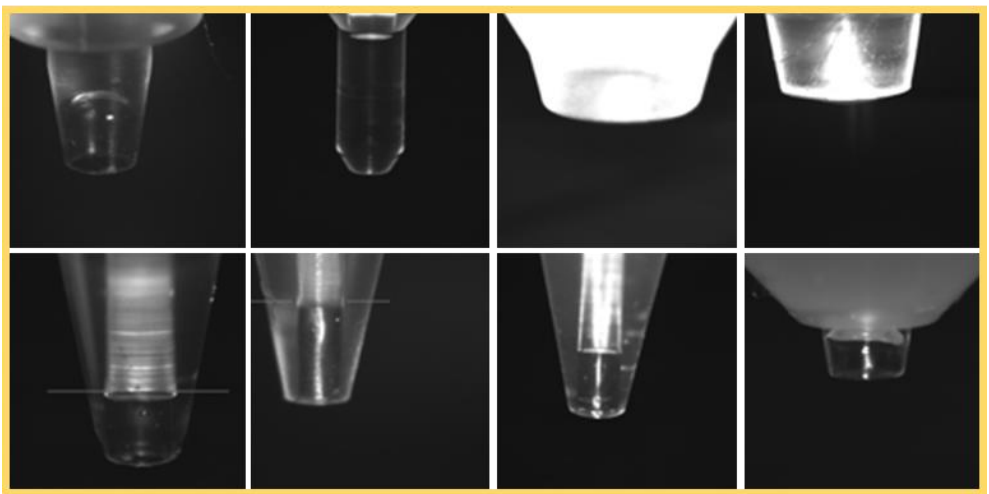


Figure 2.4 An example of various types of nozzle for spraying photoresist.

Figure 2.3 also illustrates an example of automatically detecting the suck-back state of nozzle by means of signal processing during the

spin coating process of semiconductor photolithography. In the spin coating process of semiconductor photolithography, various types of nozzle for spraying photoresist are used depending on the kind of photoresist and the characteristic of wafer. Figure 2.4 illustrates an example of various types of nozzle. Therefore, digital image processing and signal processing method used in the computer vision system should be applied to the specialized technique depending on external environment such as various types of nozzle, wafer characteristics and diffuse reflection of light etc. If a new nozzle or a new wafer is used, the defect detection accuracy of the computer vision system is inevitably reduced.

Considering these problems, we propose image segmentation method based on generalized deep learning in order to be more robust to the external environment and further improve performance instead of the specialized digital image processing and signal processing method used for semiconductor photolithography inspection.

2.2. Instance Segmentation

Object recognition contains detection and classification, which is the basis of video-based ITS [52]. Classical video based vehicle detection methods mainly include frame difference, optical flow, background subtraction based on Gaussian mixture model (GMM)

[52]. And the methods for vehicle classification usually use global or local features such as scale-invariant feature transform (SIFT), pyramid histogram of oriented gradients (PHOG), Gabor features, Harris corner [53]. In recent years, a series of famous region-based convolutional neural networks (CNNs) are proposed for simultaneously detection and classification, which have higher precision compared with the traditional methods. Girshick et al. [5] combine region proposals with CNNs to generate regions with CNN features (R-CNN). In R-CNN, candidate regions are produced through selective search [54]. Nevertheless, the inference of R-CNN is time-consuming because the CNNs is repeatedly applied for about 2000 regions per image. For improving the performance, Spatial Pyramid Pooling Network (SPP-net) is introduced by He et al. [55]. In SPP-net, the feature maps of an input image are computed only once. In order to further increase training and inference speed and improve detection precision, Fast R-CNN is developed [6]. However, Fast R-CNN still adopts the method like selective search in the step of generating the proposal regions, which is time-consuming. Ren et al. [7] propose a region proposal network (RPN) to efficiently generate proposal regions that are classified and regressed by Fast R-CNN, and the entire network composed of RPN and Fast R-CNN is called Faster R-CNN. By adding a branch for segmentation in Faster R-CNN and enhancing the features of input image through FPN [56], Mask R-CNN is proposed [8], which is used to recognize the objects.

There are two types of typical frameworks for instance segmentation principally: the proposal-based method and the proposal-free method. The former detects bounding boxes first and then refines the instances to pixel-wise masks, while the latter directly generates instance masks with no need for proposals. Currently, most of popular and advanced methods for instance segmentation use proposals, which usually include multiple stages [57-59].

Proposal-free methods belong to another stream for instance segmentation [60, 61]. Proposal free network (PFN) [62] tends to predict instance numbers and location vectors, and then clusters pixels into instances. InstanceCut [63] feeds both semantic segmentation scores and instance boundary scores into an image partition module for mask prediction. Deep Watershed Transform [64] performs an energy map learning, which is followed by a cutting operation based on a single energy level, to yield instances from related components. Sequential grouping networks (SGN) [65] employs a sequence of networks to gradually compose objects from pixels to lines and eventually to instances. Arnab et al. [66] design a dynamically instantiated conditional random field (CRF) network to predict instances mask, where they also utilize box information. Liu et al. [67] present an instance segmentation scheme via affinity derivation and graph merge, where the former estimates pixel-wise similarity and the latter merges instances with a constructed affinity graph. Single-shot instance segmentation with affinity pyramid

(SSAP) [68] also learns pixel affinity but produces affinity pyramid instead, then performs the cascade graph partition to merge instances. SSAP achieves state-of-the-art performance of proposal-free methods on Cityscapes [69]. Very recently, YOLACT [70] devotes to a real-time instance algorithm implementation, which is based on an object detection framework with lighter box prediction heads and a mask coefficient prediction head. Xu et al. [71] propose another method, i.e., explicit shape encoding for real-time instance segmentation (ESE-Seg), toward real-time instance segmentation, which predicts the boundary of instances by shape encoding directly. With Chebyshev polynomials, ESE-Seg approximates the shape coefficients of the instance inside the bounding box, thus greatly reduces the computational consumption. PolarMask [72] formulates the instance segmentation problem as instance center classification and dense distance regression in a polar coordinate, extending a similar scheme with fully convolutional one-stage object detection (FCOS) [73]. The novel polar loU loss benefits dense distance regression of instances. Proposal-free is another important method for instance segmentation, which also has succeeded in instance segmentation [65], [67], [68], [74], [75], [76]. Proposal-free does not need a proposal box prepared in advance, so it won't be influenced by bounding boxes or region proposals' accuracy. Instead, proposal-free usually exploits the spatial or semantic relation between pixel and pixel, or pixel and instance to find instance and segment instance. Besides, due to the lack of a one-stage or two-

stage object detection process, proposal-free methods are usually faster but fail on the performance compared with a proposal-based method. Neven et al. [75] proposes the loss function to pull the spatial embedding of pixels together, which belong to the same instance. This method of clustering is representative of proposal-free. To cluster pixels into instances, Liu et al. [67] proposes a graph merge algorithm that regards pixels as the vertexes and affinities as edges. By jointly learning the affinity pyramid and the semantic class labeling to compute the probability that two pixels belong to the same instance in a hierarchical manner, the performance of SSAP [68] can complete with proposal-free methods. Without instance wise labeling, Inter-pixel Relation Network (IRNet) [74] aims to learn and predict semantic affinities between pixels with image-level supervision through class boundary detection to achieve instance segmentation. These proposal-free methods achieve remarkable performance in instance segmentation. However, because of lacking a localization step compared with proposal-based methods, there is still a performance gap between proposal-free and proposal-based methods.

Proposal-based is the most dominant method of instance segmentation and achieves outstanding performance. It usually gets bounding boxes by object detector and segments these bounding boxes to get instance masks. If the object detector cannot predict an accurate bounding box, the part of instance beyond the bounding boxes cannot be segmented. Currently, the most dominant methods

[8], [77], [78], [79] for instance segmentation are proposal-based, which need to get bounding boxes or region proposals first, then segment the instance within the bounding box. Mask R-CNN [8] is the most representative work, which extends Faster R-CNN [7] by adding a fully convolutional network for instance segmentation, in parallel with the existing branch for object detection. With the instance-first strategy, the forward stream of the most representative proposal-based instance segmentation framework, Mask R-CNN [8], is: (i) region proposal network (RPN) [7] is employed to propose possible object regions; (ii) after that, the proposals are fed into the subsequent detection head to classify the category and regress accurate position coordinate offsets; (iii) at last, fully convolutional networks (FCN) [41] is utilized on the feature maps inside the bounding boxes to label the pixel-wise instance masks. Because of its great success, many other state-of-the-art methods adopt a similar framework. For instance, PANet [77] is built on Mask R-CNN, which improves information paths and aggregates features to achieve enhanced performance. DetNet [80] is an effective backbone network designed for object detection, which also benefits instance segmentation task. MaskLab [81] shows comparable performance with other state-of-the-art methods with the help of fused features from two extra outputs, namely, semantic segmentation and instance center direction. Furthermore, based on Mask R-CNN [8], Mask Scoring R-CNN [78] adds a Mask IOU head to predict a quality score for instance mask to take the place of

confidence score and achieves competitive performance. Compared with the top-down path in FPN [56], PANet [77] creates a bottom-up path augmentation to enhance the feature pyramid and shorten the information path, which improves the instance segmentation performance based on Mask R-CNN [8]. Extending on Cascade R-CNN [82], HTC [79] successes in instance segmentation by adding a semantic segmentation branch to enrich spatial context information while enhancing the information flow between each mask branch in different stages. YOLACT [70] and CenterMask [83] are respectively developed an instance segmentation sub-network on RetinaNet [32] and FCOS [73]. These methods extend on a one-stage object detector, achieve fast instance segmentation, and succeed in performance. However, these successful instance segmentation methods rely on the accuracy of object detection incredibly. If the object detector cannot predict an accurate bounding box, instance segmentation's performance will degenerate. Instance segmentation requires both pixel-level classification accuracy and high-level semantic features at the target instance level, which is very challenging, and the cascade structure can effectively improve both of these problems. To make full use of the relationship between detection and segmentation, Wen et al. [84] proposed a joint multi-tasking cascade structure, which is not simply to cascade the two tasks of detection and segmentation, but to unitedly put them into multi-stage processing, and especially to integrate the information at different stages of the mask branch. In addition, the feature fusion

process is introduced in the full convolution networks (FCN) branch, and the high-level and low-level features are effectively fused to enhance the contextual information of the picture semantic features. In order to create a more accurate instance segmentation, various improved and extended methods based on mask R-CNN have been presented. Zhang et al. [85] designed the segmenting beyond the bounding box (S3B-Net) extended on Mask R-CNN to help instance segmentation methods based on object detection to segment the part of an instance beyond the bounding box. Specifically, the sub-network first predicts a two-dimensional pixel embedding for each pixel. Then, the Gaussian function is employed to calculate a pixel's probability belongs to a corresponding instance according to the two-dimensional pixel embedding. Finally, the output of the sub-network combines with the output of instance segmentation based on object detection to generate a more precise instance mask. Wen et al. [86] proposed an automatic building extraction method based on improved mask region convolutional neural network method that can detect the rotated bounding boxes of buildings and segment them from very complex backgrounds, simultaneously. In order to tackle the ambiguity in the acquired outdoor depth map, Xu et al. [87] proposed a residual regretting mechanism, incorporated into current flexible, general and solid instance segmentation framework Mask R-CNN in an end-to-end manner. Specifically, regretting cascade is designed to gradually refine and fully unearth useful information in depth maps, acting in a filtering and backup way. Additionally,

embedded by a novel residual connection structure, the regretting module combines RGB and depth branches with pixel-level mask robustly.

Instance segmentation based on Mask R-CNN has a wide range of application scenarios. Zhang et al. [88] presented that a traffic surveillance system for obtaining comprehensive vehicle information, including type, speed, length, current driving lane, and traffic volume, is proposed based on instance segmentation which is realized by Mask R-CNN. Chen et al. [89] presented a novel method based on Mask R-CNN to estimate building areas in property assessment. This method is to fine-tune an initial model obtained from transfer learning with a small number of drone aerial images. Our approach also is to use proposal-based Mask R-CNN framework for inspection of semiconductor photolithography process.

2.3. Backbone Structure

Recent years have witnessed numerous backbone networks [90], [91], [92], [93], [94], [95], [96], achieving state-of-the-art performance in various vision tasks with stronger multiscale representations. As designed, CNNs are equipped with basic multi-scale feature representation ability since the input information follows a fine-to-coarse fashion. The AlexNet [92] stacks filters

sequentially and achieves significant performance gain over traditional methods for visual recognition. However, due to the limited network depth and kernel size of filters, the AlexNet [92] has only a relatively small receptive field. The VGGNet [93] increases the network depth and uses filters with smaller kernel size. A deeper structure can expand the receptive fields, which is useful for extracting features from a larger scale. It is more efficient to enlarge the receptive field by stacking more layers than using large kernels. As such, the VGGNet [93] provides a stronger multi-scale representation model than AlexNet [92], with fewer parameters. However, both AlexNet [92] and VGGNet [93] stack filters directly, which means each feature layer has a relatively fixed receptive field. Network in Network (NIN) inserts multi-layer perceptron as micro-networks into the large network to enhance model discriminability for local patches within the receptive field. The 1×1 convolution introduced in NIN has been a popular module to fuse features. The GoogLeNet [94] utilizes parallel filters with different kernel sizes to enhance the multi-scale representation capability. However, such capability is often limited by the computational constraints due to its limited parameter efficiency. The Inception Nets [97], [98] stack more filters in each path of the parallel paths in the GoogLeNet [94] to further expand the receptive field. On the other hand, the ResNet [90] introduces shortcut connections to neural networks, thereby alleviating the gradient vanishing problem while obtaining much deeper network structures. During the feature extraction procedure,

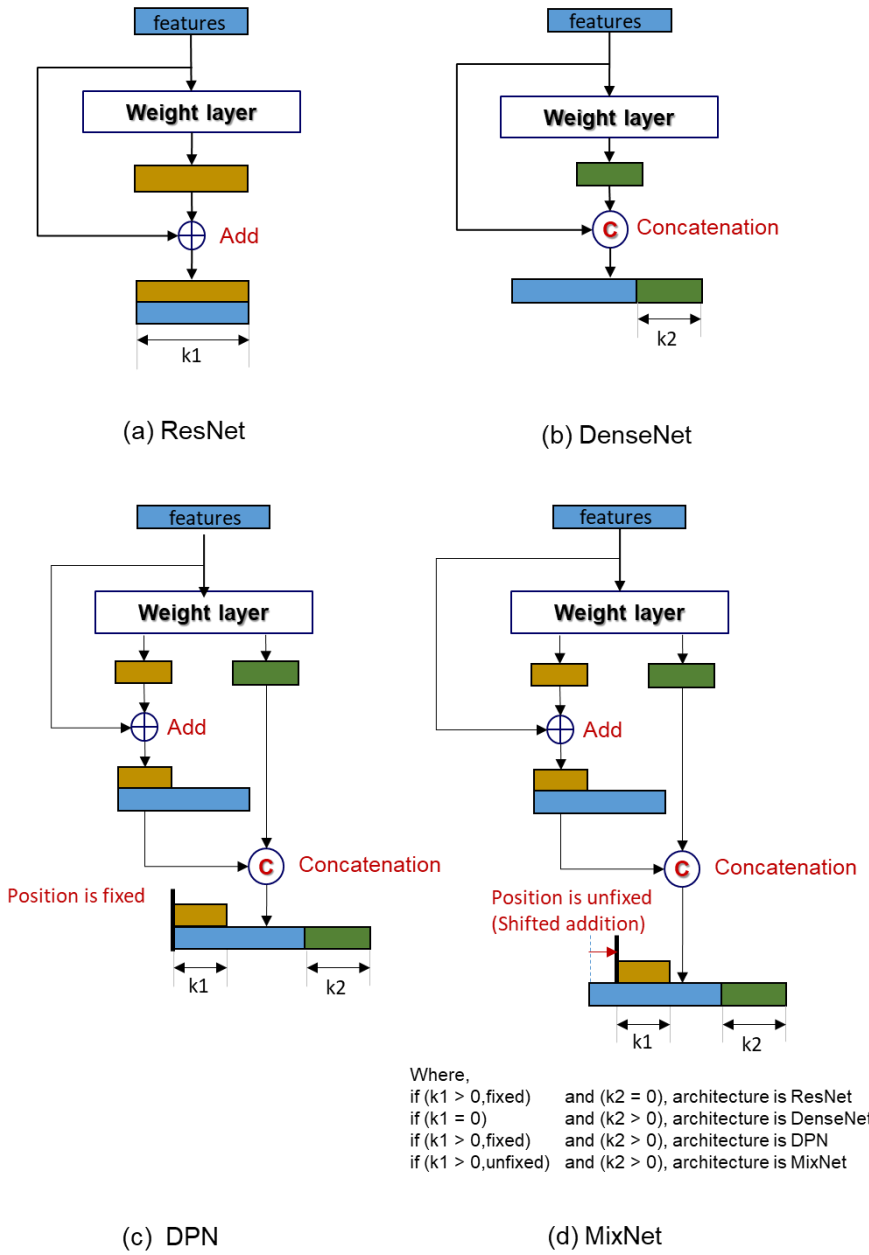


Figure 2.5 The structure of four architectures. The vertically aligned features are merged by element-wise addition, and the horizontally aligned features are merged by concatenation. (a), (b) and (c) are the three derivative architectures with various representative settings of (d).

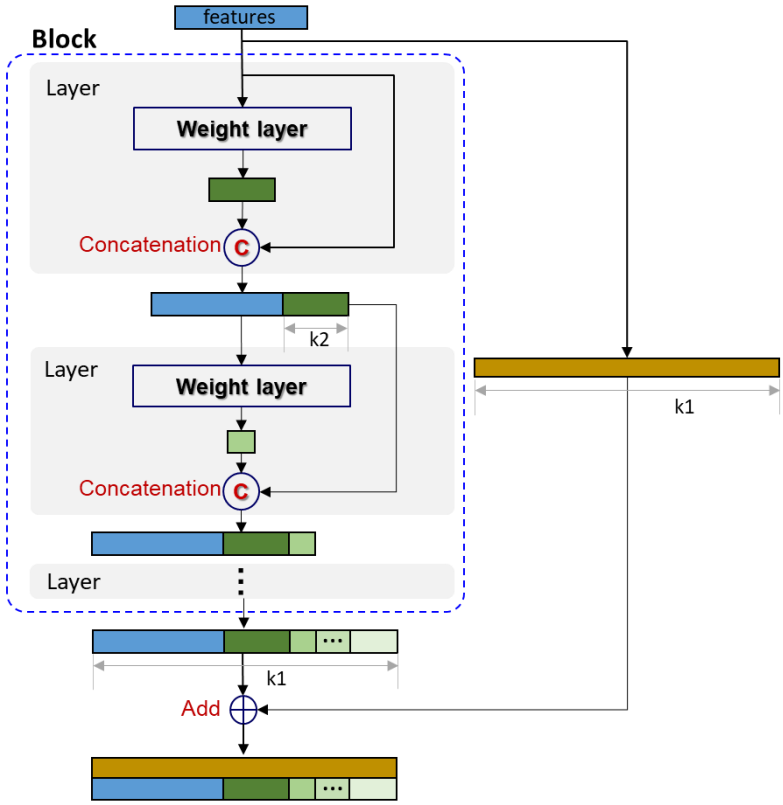


Figure 2.6 The structure of our architecture. The vertically aligned features are merged by element-wise addition, and the horizontally aligned features are merged by concatenation. The k1 is the same size as the output of the block. The k2 is dynamic growth rate which is different in each layer.

shortcut connections allow different combinations of convolutional operators, resulting in a large number of equivalent feature scales. Shortcut connection is a connection that skips one or more layers. In case of ResNet, as shown in Figure 2.5 (a), the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Similarly, densely connected layers in the DenseNet [91] enable the network to process objects in a very

wide range of scales. DPN [99] combines the ResNet with DenseNet to enable feature re-usage ability of ResNet and the feature exploration ability of DenseNet. Furthermore, Wang et al. [100] presents Mixed Link Network (MixNet) which is a more generalized form than other existing modern networks (ResNet, DenseNet and DPN). It can be seen from Figure 2.5 that the mixed link architecture (Figure 2.5 (d)) with different parametric configurations can reach three representative architectures (Figure 2.5 (a) (b) (c)). Our architecture has the same concept of combining feature re-usage ability of ResNet and feature re-exploration ability of DenseNet as in DPN and MixNet. However the structure is significantly different as shown in Figure 2.6. Notably, in order to achieve superior efficiency with compactness, we combine feature re-usage with same size in unit of block which is a group of layers and feature re-exploration with dynamic growth rate in unit of layer. Additionally, the recently proposed DLA [96] method combines layers in a tree structure. The hierarchical tree structure enables the network to obtain even stronger layer-wise multi-scale representation capability.

In object detection, the backbone acts as the main feature extractor, which takes images or videos as input and yields corresponding feature maps [10]. According to the specific needs of detection accuracy and efficiency, different backbones can be developed for a model after modification or tuning. For high accuracy, a deep and densely connected backbone, such as the ResNet and

DenseNet, can be employed in the model. Considering the speed and efficiency, lightweight backbones, such as the MobileNets and EfficientNet, would be preferred.

In practice application, to accurately identify multiple railroad track components, Guo et al. [101] proposed and evaluated YOLACT-Res2Net-50 and YOLACT-Res2Net-101, which adapt a new backbone architecture compared to the original models. ResNet-50 and ResNet-101 backbone [90] are adopted in the original YOLACT models. As the name indicates, ResNet-50 and ResNet-101 include 50 layers and 101 layers, respectively. To reduce the inference computations, the bottleneck structure is introduced in the ResNet. With the bottleneck design for ResNet-50 and ResNet-101, the first 1×1 convolution reduces a 256 dimension channel to a 64-dimension channel, and it is recovered by a 1×1 convolution at the end. Res2Net [102] is a new backbone architecture which can improve the multi-scale representation capability at a granular level. The architecture of the Res2Net bottleneck plays an important role in the new backbone. In this bottleneck structure, the original 3×3 filter of n channels is replaced with a set of smaller filter groups.

Backbone network is acting as the basic feature extractor for object detection task which takes images as input and outputs feature maps of the corresponding input image. Most of backbone networks for detection are the network for classification task taking out the last fully connected layers. The improved version of basic

classification network is also available. For instance, Lin et al. [103] add or subtract layers or replace some layers with special designed layers. Redmon et al. [104] proposed extremely fast network which is inspired by the GoogLeNet model for image classification. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

There has been little work discussing on the backbone feature extractor specifically designed for the object detection. More importantly, there are several differences between the tasks of image classification and object detection. (i) Recent object detectors like FPN and RetinaNet usually involve extra stages against the task of image classification to handle the objects with various scales. (ii) Object detection not only needs to recognize the category of the object instances but also spatially locate the position. Large down sampling factor brings large valid receptive field, which is good for image classification but compromises the object location ability. Due to the gap between the image classification and object detection, Li et al. [105] proposed a novel backbone network specifically designed for object detection. Moreover, the network includes the extra stages against traditional backbone network for image classification, while maintains high spatial resolution in deeper layers.

Due to the advancement of deep learning, classification accuracy has improved greatly. However, conversely, the complexity of the

model has also increased accordingly. As a consequence, researches aiming to maintain accuracy as much as possible while reducing the number and size of parameters of the model are also presented. Towards different requirements about accuracy vs. efficiency, we can choose deeper and densely connected backbones. Xie et al. [95] proposed a simple, highly modularized network architecture which is constructed by repeating a building block that aggregates a set of transformations with the same topology. By repeatedly constructing the same block, image classification is possible with fewer parameters, and increasing the cardinality which is the size of the set of transformations rather than a deeper and wider dimension improves the accuracy of classification. G.Howard et al. [106] proposed a class of efficient models called MobileNets for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions. MobileNets use depthwise convolutions to apply a single filter per each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output of the depthwise layer. MobileNets use both batchnorm and ReLU nonlinearities for both layers. Depthwise convolution is extremely efficient relative to standard convolution. However, it only filters input channels, it does not combine them to create new features. So an additional layer that computes a linear

combination of the output of depthwise convolution via 1×1 convolution is needed in order to generate these new features. Sandler et al. [107] introduces a new neural network architecture that is specifically tailored for mobile and resource constrained environments. The network pushes the state-of-the-art for mobile tailored computer vision models, by significantly decreasing the number of operations and memory needed while retaining the same accuracy. The network has a novel layer module: the inverted residual with linear bottleneck. This module takes as an input a low-dimensional compressed representation which is first expanded to high dimension and filtered with a lightweight depthwise convolution. Features are subsequently projected back to a low-dimensional representation with a linear convolution. The inverted residual connections is a combination of depthwise separable convolution and linear bottleneck.

In order to achieve better performance than MobileNets while reducing the amount of computation, Zhang et al. [108] proposed the new architecture that utilizes two new operations, pointwise group convolution and channel shuffle, to greatly reduce computation cost while maintaining accuracy. Grouped Convolution is a method of independently performing a convolution operation by dividing the channel of an input value into several groups. It is simple to implement and is advantageous for parallel processing. In order to reduce side effects due to group convolution, information can be exchanged for each channel through an operation called channel

shuffle. First of all, ShuffleNet basically uses the structure of MobileNets. The 1×1 convolution takes up almost all of the computation. It tries to further reduce the 1×1 convolution according to this amount of computation. It's a grouped convolution. In the 1×1 convolution layer, not all channels are considered, but only some of the channels are considered and the channels are mixed in the middle to consider all channels.

Furthermore, Iandola et al. [109] proposed a small CNN architecture called SqueezeNet which achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters. SqueezeNet uses 8 fire modules and 1 convolutional layer each at the I/O stage. At this time, the fully connected layer was not used at all, because the fully connected layer has a fairly large amount of parameters, and if you match this too large amount of parameters, the probability of overfitting is increased. Therefore, SqueezeNet can be free from the problem of overfitting by using Global Average Pooling instead of fully connected layer. In addition, to focus on speed, Chollet [110] proposed the Xception architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolution operation, which is a depthwise convolution followed by a pointwise convolution.

In practice, network models used in the real-world require more lightweight since the device to be used is small and requires a short inference time. When applied to mobile devices, lightweight backbones can meet the requirements. Wang et al. [40] propose a

novel real-time object detection system by combining PeleeNet with SSD [31] and optimizing the architecture for fast processing speed. Furthermore, Wang et al. [111] propose Cross Stage Partial Network (CSPNet) to mitigate the problem that previous works require heavy inference computations from the network architecture perspective. CSPNet reduces the duplicate gradient information within network optimization to be light-weighted for mobile GPUs or CPUs. In order to meet the needs of high precision and more accurate applications, complex backbones are needed. On the other hand, real-time acquirements like video or webcam require not only high processing speed but high accuracy YOLO model [104], which need well designed backbone to adapt to the detection architecture and make a trade-off between speed and accuracy. To explore more competitive detecting accuracy, deeper and densely connected backbone is adopted to replace the shallower and sparse connected counterpart. He et al. [8] utilize ResNet [90] rather than VGG [93] to capture rich features which is adopted in Faster R-CNN [7] for further accuracy gain because of its high capacity. The newly high performance classification networks can improve precision and reduce the complexity of object detection task. This is an effective way to further improve network performance because the backbone network acts as a feature extractor. As is well known to all, the quality of features determines the upper bound of network performance, thus it is an important step that needs further exploration. Likewise, our goal is to design a simple yet resource efficient network structure

with keeping high detection accuracy for image segmentation to be applied in semiconductor photolithography inspection systems.

2.4. Enhanced Feature Map

The discriminative feature is very important factor in image classification problem, and the smaller the variance within the same class and the larger the variance between different classes, the easier it is to solve the classification problem in general.

The improved network structure for enhanced feature map is shown in Figure 2.7. The nozzle image is input into the network, and then different feature maps are output by means of a series of convolution and pooling in feature pyramid networks (FPN). After that, different feature maps are delivered into the region proposal networks (RPN) so as to extract the region of interest (ROI). Then the ROI is input to the ROI align to perform pixel correction on the feature map for subsequent target classification and bounding box regression. In the mask branch, the original images are cropped using the corrected bounding box, and then the images in ROI are performed by mask prediction.

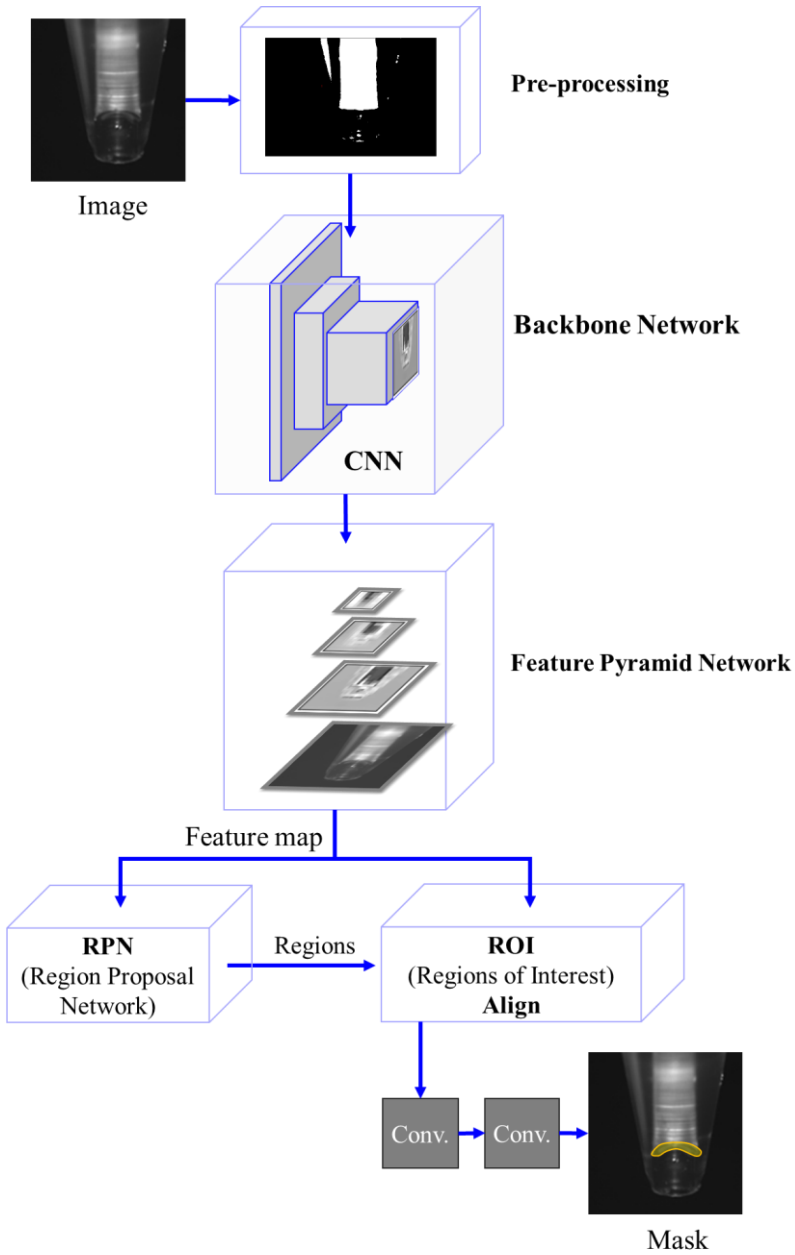


Figure 2.7 The improved network structure for enhanced feature map.

Pre-processing. Various researchers have shown the importance of network architecture in achieving better performances by making changes in different layers of the network. Kuntal Kumar Pal et al. [112] have shown the importance of pre-processing techniques for

image classification using the CIFAR10 dataset and three variations of the Convolutional Neural Network. Jonghwa Yim et al. [113] propose a generalized architecture of a dual channel model to treat quality degraded input images. The dual-channel architecture with two inputs has the original model using unprocessed image and the augmented model using de-noised image by means of pre-processing. It is necessary to use image enhancement technology to enhance the contrast of input images and to highlight the features in the image. In order to better deal with local features, Jiangping Qin at el. [114] uses the contrast limited adaptive histogram equalization (CLAHE) algorithm to pre-process images [115]. Most of the contrast enhancement techniques are based on histogram modifications, which can be performed globally or locally.

As illustrated in Figure 2.8, the contrast limited adaptive histogram equalization (CLAHE) is a method which can overcome the limitations of global approaches by performing local contrast enhancement. However, this method relies on two essential hyper parameters: the number of tiles and the clip limit. An improper hyper parameter selection may heavily decrease the image quality toward its degradation. Campos et al. [115] proposed the LB-CLAHE: a learning based hyper parameter selection method for CLAHE using image features.

From this study, we conclude that pre-processing raw image data achieves the best performance for convolutional neural network and the performance increases even more with the increase in

convolutional layers in the architecture. If we use well-known image processing algorithm as a pre-processing module to emphasize the outlines and edges and as a method to reinforce the specific features or we design a customized preprocessing module, better results are expected on performance.

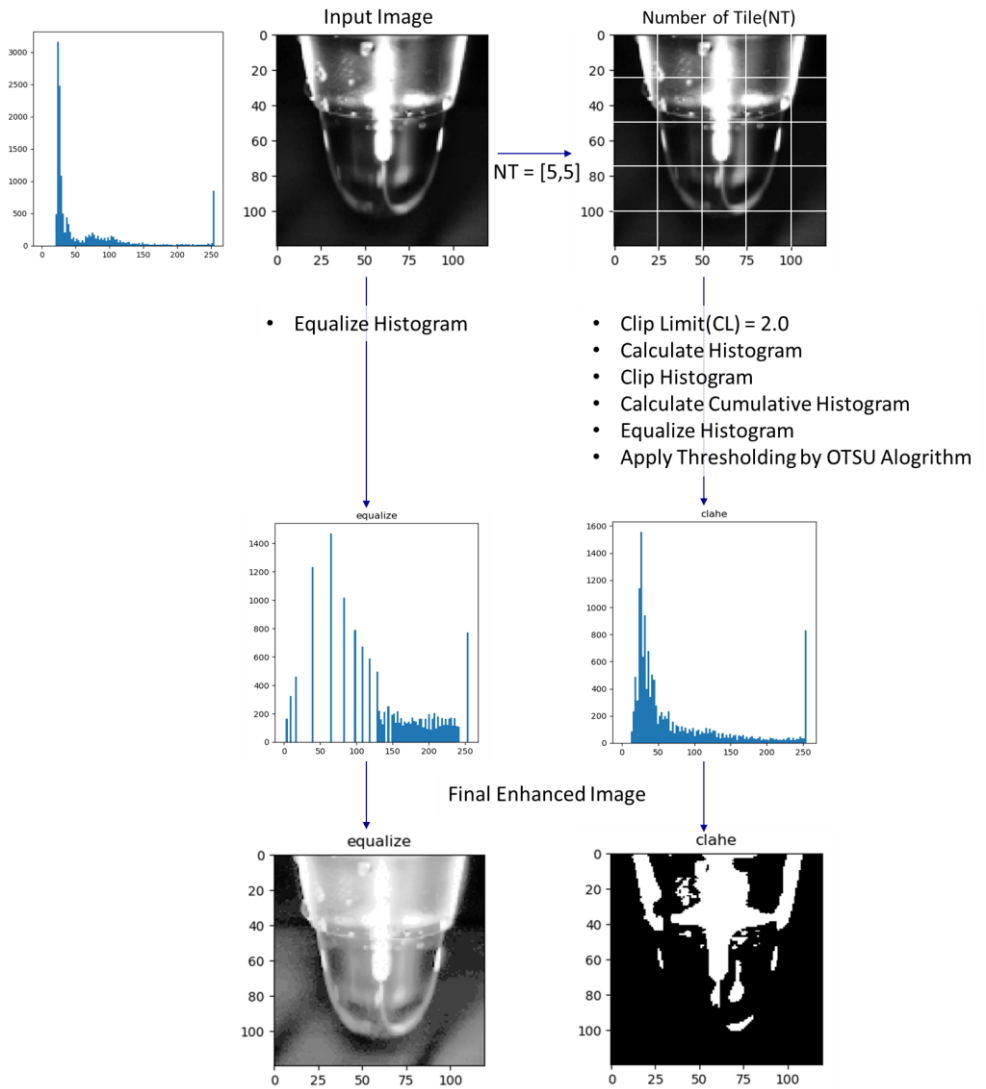


Figure 2.8 Overview of CLAHE. Example with Number of Tiles(NT) = [5,5] and Clip Limit(CL) = 2.0.

Backbone Network. A lot of deep convolutional neural networks (CNN) [92] originally designed for classification tasks have been adopted for the detection task as well. And a lot of modifications have been done on them to adapt for the additional difficulties encountered. Object detection is a natural extension of the classification problem. The constant challenge is to correctly detect the presence and accurately locate the object instance in the image. It is a supervised learning problem in which, given a set of training images, one has to design an algorithm which can accurately locate and correctly classify as many object instances as possible in a rectangle box while avoiding false detections of background or multiple detections of the same instance. The process of detecting instance segmentation can be spilt into three parts: extracting feature-maps, proposing regions, classifying and regressing binary mask. As aforementioned, among them, the backbone network that extracts feature-maps is crucial to a major role in instance segmentation detection models. Huang et al. [116] partially confirmed the common observation that, as the classification performance of the backbone increases on ImageNet [23] classification task, so does the performance of object detectors based on those backbones. It is the case at least for popular object detectors like Fast R-CNN [6], Faster R-CNN [7], Mask R-CNN [8] and R-FCN [30] although for SSD [31] the object detection performance remains around the same. Since there are significant efforts that have been devoted to design network architectures for image classification, many diverse and powerful networks are

emerged, such as VGGNet [93], GoogLeNet [94] , ResNet [117], DenseNet [91], DPN [99] etc. In practice, most of the detection methods [5],[6],[7], [8],[31] directly utilize these structures pre-trained on ImageNet [13] as the backbone network for detection task. Some other works try to design specific backbone network structures for object detection, but still require to pre-train on ImageNet [23] classification dataset in advance. Kim et al. [118] proposes PVANet for fast object detection, which consists of the simplified "Inception" block from GoogLeNet [94]. Huang et al. [116] investigated various combination of network structures and detection frameworks, and found that Faster R-CNN [7] with Inception-ResNet-v2 [97] achieved very promising accurate performance. Nakazawa et al. [119] proposed the CNN architecture for wafer map pattern generation in the semiconductor manufacturing.

Therefore, we present a suitable backbone structure for extracting the enhanced feature-map to detect image segmentation in industrial domain, which is the proposed image segmentation detector (ISD) instead of ResNet [117] that is the backbone network of state-of-the-art Mask R-CNN framework.

Feature Pyramid Network. Extracting effective features from input images is a vital prerequisite for further accurate classification and localization steps. To fully utilize the output feature maps of consecutive backbone layers, Lin et al. [56] presented feature pyramid network (FPN) which is a multi-scale feature fusion network structure to extract richer features by dividing them into

different levels to detect objects of different sizes. As illustrated in Figure 2.9, FPN is different from the traditional image pyramid structure. It is divided into three parts: bottom-up, top-down, and horizontal-connection. The bottom-up pathway is the feedforward computation of the backbone ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2. The top-down pathway hallucinates higher resolution features by up sampling spatially coarser, but semantically stronger, feature maps from higher pyramid levels.

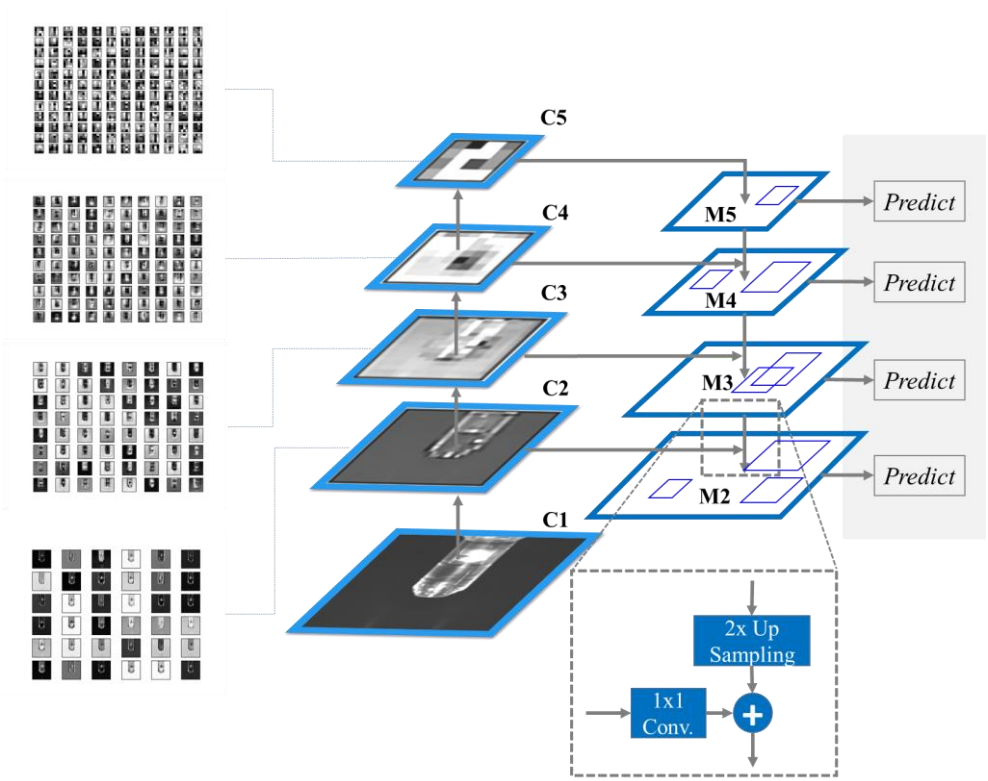


Figure 2.9 A feature pyramid with predictions made independently at all levels. A building block illustrating the lateral connection and the top-down pathway, merged by addition.

These features are then enhanced with features from the bottom-up pathway via lateral connections. Each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway. The bottom-up feature map is of lower-level semantics, but its activations are more accurately localized as it was subsampled fewer times.

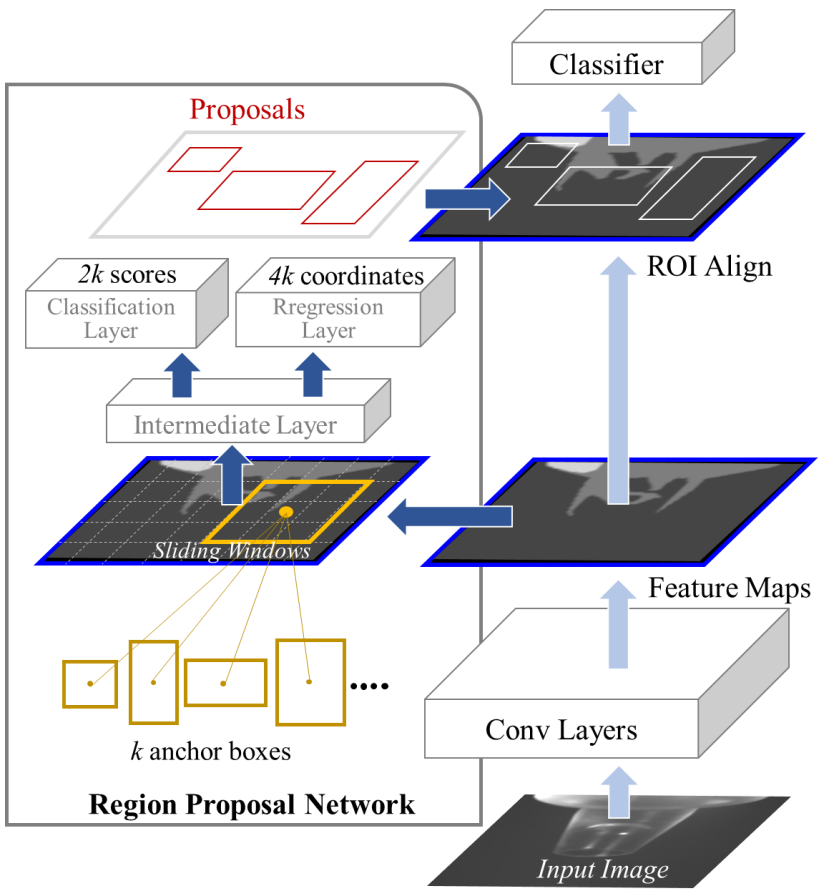


Figure 2.10 Region Proposal Network (RPN). The RPN module serves as the "attention" of single unified network. In other words, The RPN module tells the network module where to look.

In order to extract more contextual semantic information of small objects, Liu et al. [120] propose Multi-branch Parallel Feature Pyramid Networks (MPFPN), which aims to extract more abundant feature information of the objects with a small size. The parallel branch is designed to recover the features that missed in the deeper layers. Furthermore, to tackle multiscale objects detection problem, Li and Zhou [121] proposed Feature Fusion SSD (FSSD) by adding a lightweight and efficient feature fusion module to the conventional SSD. In the feature fusion module, features from different layers with different scales are concatenated together, followed by some down-sampling blocks to generate new feature pyramid, which will be fed to multi-box detectors to predict the final detection results.

Region Proposal Network. A Region Proposal Network (RPN) takes an image of any size as input and outputs a set of rectangular object proposals, each with an objectness score by the team of Shaoqing Ren [7]. The RPN shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. As illustrated in Figure 2.10, The RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. At each sliding-window location, the RPN simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k. The k proposals are parameterized relative to k reference boxes, which is called as anchors. An anchor is centered at the sliding window.

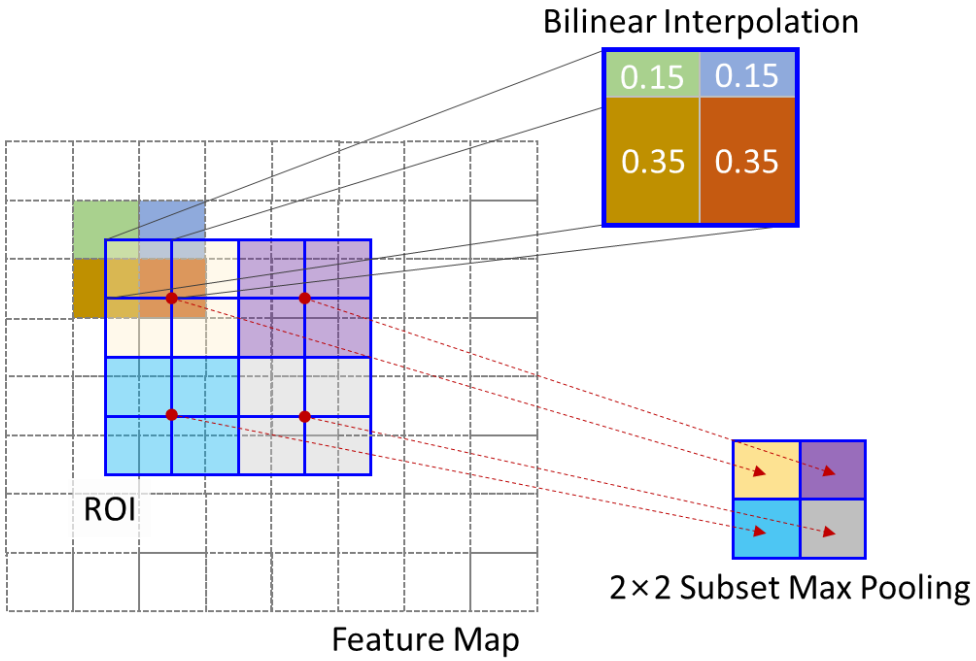


Figure 2.11 ROI align structure. ROI pooling was a model for object detection. It wasn't important to have accurate location information. If the ROI has floating point coordinates, the floating point is rounded and then pooling is done. This distorts the location information of the input image, causing segmentation problems. Therefore, ROI alignment using positional information is used using bilinear interpolation.

So the a regression layer has *4k* outputs encoding the coordinates of k boxes, and the classification layer outputs *2k* scores that estimate probability of object or not object for each proposal.

Region of Interest Align. The ROI align method was proposed in Mask R-CNN. Because it does not perform the quantization and rounding of coordinates of the ROI area, the problem of misalignment between the feature-map and the original image in ROI pooling was solved by ROI align. The structure of ROI align is shown in Figure

2.11. The region with dotted line represents the generated feature-maps, and the rectangle region surrounded by a solid line represents the ROI that has been adjusted. The ROI is divided into 4×4 cells. If the number of samples in each cell is 4, each cell will be averaged divided into four bins, and the center of each bin is the sampling point. Since the coordinates of the ROI are floating-point numbers, the coordinates of the sampling points are usually also floating-point numbers. Therefore, bilinear interpolation is adopt for each sampling point pixel. This operation can be used to obtain the pixel value of the sampling point, and then four sampling points are performed max pooling on each cell. Finally, the ROI align output are obtained

The feature-map of CNN to detect nozzle type is clearly distinguished between the nozzle types. However, since the inspection in semiconductor photolithography is performed in the same nozzle type, it is difficult to extract the discriminative CNN feature-map. It is hard to extract the discriminative feature from the proposed regions of the Region Proposal Network (RPN) using FPN feature-map of the same nozzle type. As illustrated in Figure 2.5, the mask area cannot be achieved without the discriminative CNN feature-map in the proposed regions.

The reason for not being able to extract the discriminative feature in the proposed regions is that it is not enough to extract the discriminative feature by means of only original pixel information in the corresponding area as a gray scale image. In order to enhance a feature-map of CNN with only the original pixel information of the

image, it may be possible to extract the discriminative feature by performing a lot of deep learning by increasing the network layer of CNN with a large number of various training images. Deep convolutional neural networks require a large corpus of training data in order to avoid over-fitting. Over-fitting refers to the phenomenon when a network learns a function with very high variance such as to perfectly model the training data. Unfortunately, many application domains do not have access to big data, such as industrial image analysis and medical image analysis, and collection of such training data is often expensive and laborious.

Data augmentation overcomes this issue by artificially inflating the training set with label preserving transformations. Recently there has been extensive use of generic data augmentation to improve CNN task performance. Data augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better deep learning models can be built using them. Data augmentations based on basic image manipulations are geometric transformation, flipping, color space, cropping, rotation, translation, noise injection, color space transformations, geometric versus photometric transformations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning [122-127]. However, we propose the pre-processing method that reduces the amount of training images and decreases the number of network layer in CNN rather than data augmentation. The specialized

image filter for the semiconductor photolithography inspection is applied to the pre-processing method in order to enhance the feature-map of CNN.

2.5. Detection Performance Evaluation

There are two important evaluation indicators for the performance of the classification problem. One is precision, which is used to evaluate how many objects are correctly identified in the result of classification. The other is Recall, which is used to evaluate how many positive examples are predicted correctly in the total positive samples. The calculation formulas for Precision P and Recall R are (1) and (2), respectively.

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \quad (1)$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \quad (2)$$

Where, True Positive (TP) means that the positive class is predicted to be positive, which is a correct detection of a ground-truth bounding box; False Positive (FP) means that the negative class is predicted to be positive, which is An incorrect detection of a nonexistent object or a misplaced detection of an existing object;

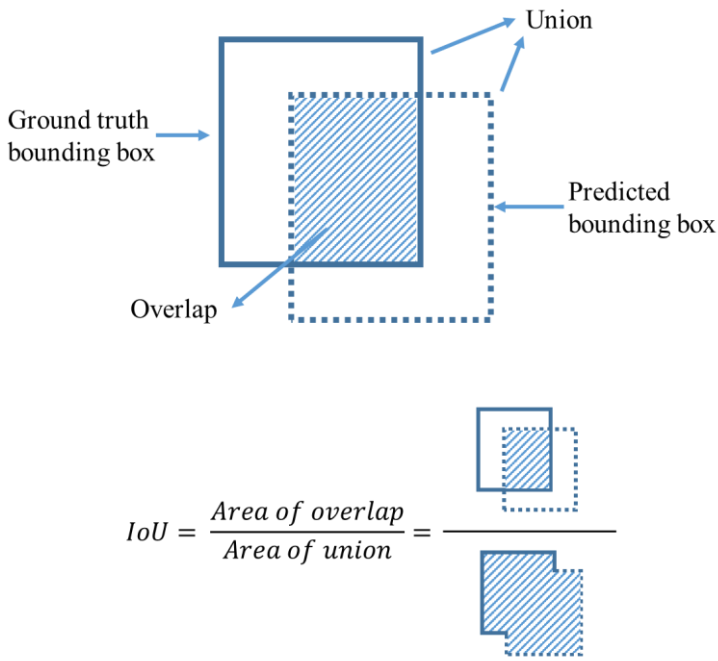


Figure 2.12 The definition of intersection over union (IoU).

False Negative (FN) means that the positive class is predicted to be negative, which is an undetected ground-truth bounding box.

For the target detection network, there is a very important concept, Intersection over Union (IoU). As illustrated in Figure 2.12, the degree of overlap of two regions is expressed by IoU. When it is adopted to test the accuracy of the network prediction, IoU expresses the overlap between the prediction box and the labeled box. The calculation formula is as follows:

$$\mathrm{IoU} = \frac{A \cap B}{A \cup B'} \tag{3}$$

In most competitions, the Average Precision (AP) and its

derivations are the metrics adopted to assess the detections and thus rank the teams. The PASCAL VOC dataset [128] and challenge [129] provide their own source code to measure the AP and the mean AP (mAP) over all object classes. The AP is evaluated with different IoUs. It can be calculated for 10 IoUs varying in a range of 50% to 95% with steps of 5%, usually reported as AP@50:5:95. It also can be evaluated with single values of IoU, where the most common values are 50% and 75%, reported as AP50 and AP75 respectively. The mean AP (mAP) is a metric used to measure the accuracy of object detectors over all classes in a specific database. The mAP is simply the average AP over all classes [7], [31], that is

$$\mathrm{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad (4)$$

Where AP_i being the AP in the ith class and N is the total number of classes being evaluated.

In this dissertation, since it has sufficient performance for inspection of semiconductor photolithography process, given that IoU is more than 50%, we adopt mAP which is evaluated with single values of IoU, where the most common values are 50% and 75%, notated as mAP@0.50 and mAP@0.75. Additionally, we adopt $\mathrm{mAP}_{\mathrm{Total}}$ that is

$$\mathrm{mAP}_{Total} = \frac{\mathrm{mAP@0.50} + \mathrm{mAP@0.75}}{2} \quad (5)$$

2.6. Learning Network Model from Scratch

There are many successful cases that fine-tuning using transfer learning works well and achieves consistent improvement, especially in object detection. So why do we still need to train our model from scratch? As stated previously, most pre-trained models are learned on large-scale RGB dataset like ImageNet. However, there is no large-scale dataset in semiconductor industry domain for transfer learning. Thus, the critical importance of training from scratch is lack of dataset in specific domain.

There are no previous works that train deep CNN-based instance segmentation in industrial domain from scratch. In generic object detection, Shen et al. [130] proposed Deeply Supervised Object Detectors (DSOD) built upon SSD, an object detection framework that can be trained from scratch. In semantic segmentation, J´egou et al. [131] demonstrated that a well-designed network structure can outperform state-of-the-art solutions without using the pre-trained models. It extends DenseNet [91] to fully convolutional networks by adding an up sampling path to recover the full input resolution.

Thus, our proposed approach has very appealing advantage in that it is learning network model from scratch without using the pre-trained model on ImageNet [23] for instance segmentation.

Chapter 3. Proposed Method

We first introduce the whole framework of our ISD architecture, following by pre-processing for extracting the enhanced feature-map. Then we describe the training process and objective, configurations in detail.

3.1. ISD Architecture

The whole framework for semiconductor photolithography inspection is based on Mask R-CNN framework. There are two stages of Mask R-CNN framework. First, it generates proposals about the regions where there might be an object based on the input image. Second, it predicts the class of the object, refines the bonding box and generates a mask in pixel level of the object based on the first stage

proposal. Both stages are connected to the backbone network structure.

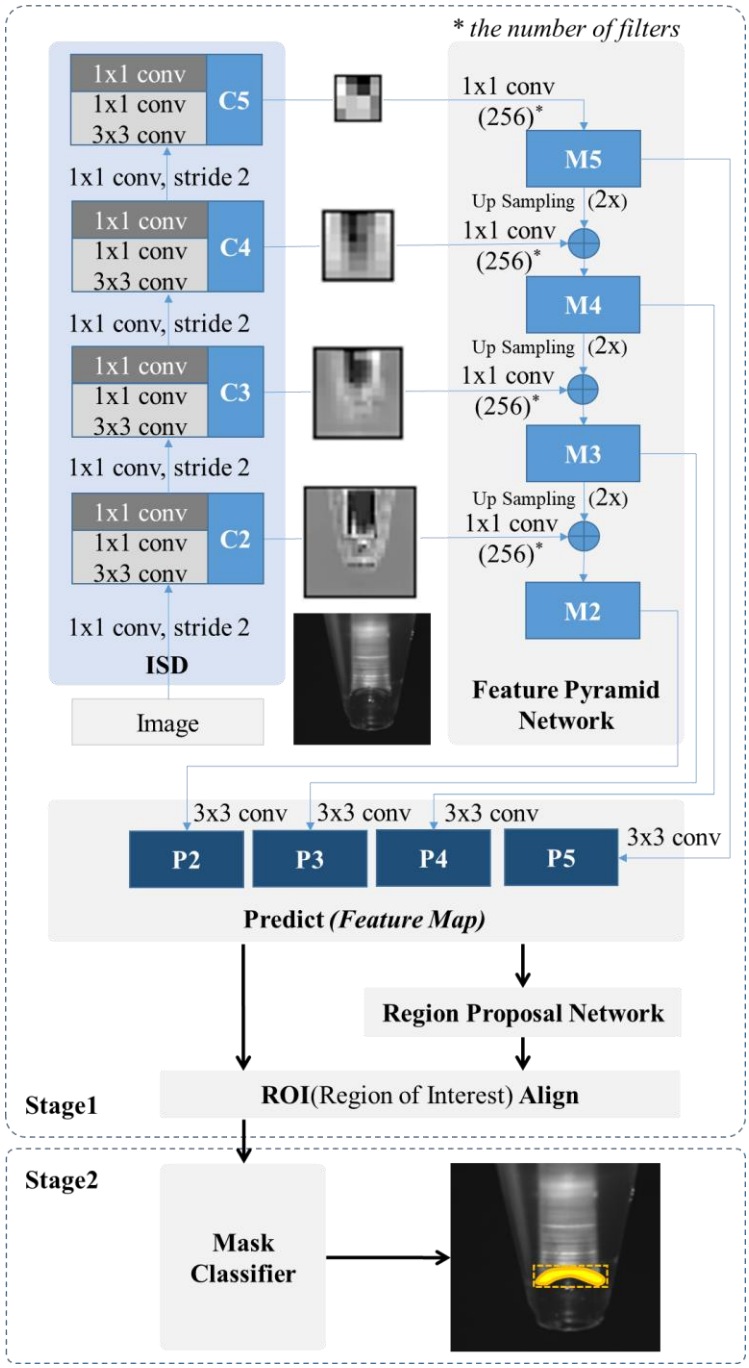


Figure 3.1 The network structure by using ISD for image segmentation on Mask R-CNN framework.

Many approaches to instance segmentation are based on segment proposals. However, our approach is focus on the backbone network which extracts the enhanced feature-maps for the object mask. The state-of-the-art Mask R-CNN framework uses ResNet [117] and ResNetXt [95] as backbone network. However, as illustrated in Figure 3.1, our approach uses the compact ISD instead of ResNet [117] for addressing real-time problem and learning from scratch. ISD based on the state-of-the-art DenseNet [91] is motivated by combining the advantage of shortcut connection and concatenation of the feature-maps produced in layers with dynamic growth rate. In order to improve the performance of instance segmentation with better parameter efficiency, we investigated all the state-of-the-art CNN based instance segmentation. The design principle of ISD is compact model, which is suitable for real-time embedded system such as computer vision system and make them easy to train under reducing over fitting on tasks with smaller training set sizes.

ISD comprises layers, each of which implements a composite function of operations such as Batch Normalization (BN) [132], rectified linear units (ReLU) [133], Pooling [134], or Convolution (Conv). ISD has the concatenation of the feature-maps produced in layers in order to encourage strengthen feature propagation and feature reuse. Further, ISD has the shortcut connection for addressing vanishing and exploding gradients. ISD is composed of four dense blocks and four transition layers similar to DenseNet [91]; see Table 3.1.

Table 3.1 ISD Architecture.

Layers	**Output Size** (input 3×120×120)	**ISD-38**
Convolution	12×60×60	3×3 conv, stride 2
Dense Block (1)	24 ×30×30	1×1 conv, stride 2
	24×30×30	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 2$
Transition Layer (1)	24×30×30	1×1 conv, stride 1
Dense Block (2)	48×15×15	1×1 conv, stride 2
	48×15×15	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 4$
Transition Layer (2)	48×15×15	1×1 conv, stride 1
Dense Block (3)	72×8×8	1×1 conv, stride 2
	72×8×8	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 4$
Transition Layer (3)	72×8×8	1×1 conv, stride 1
Dense Block (4)	96×4×4	1×1 conv, stride 2
	96×4×4	$\begin{bmatrix}1\times1\text{ conv}\\3\times3\text{ conv}\end{bmatrix}\times 4$
Transition Layer (4)	96×4×4	1×1 conv, stride 1
Prediction	-	Pooling/Dense

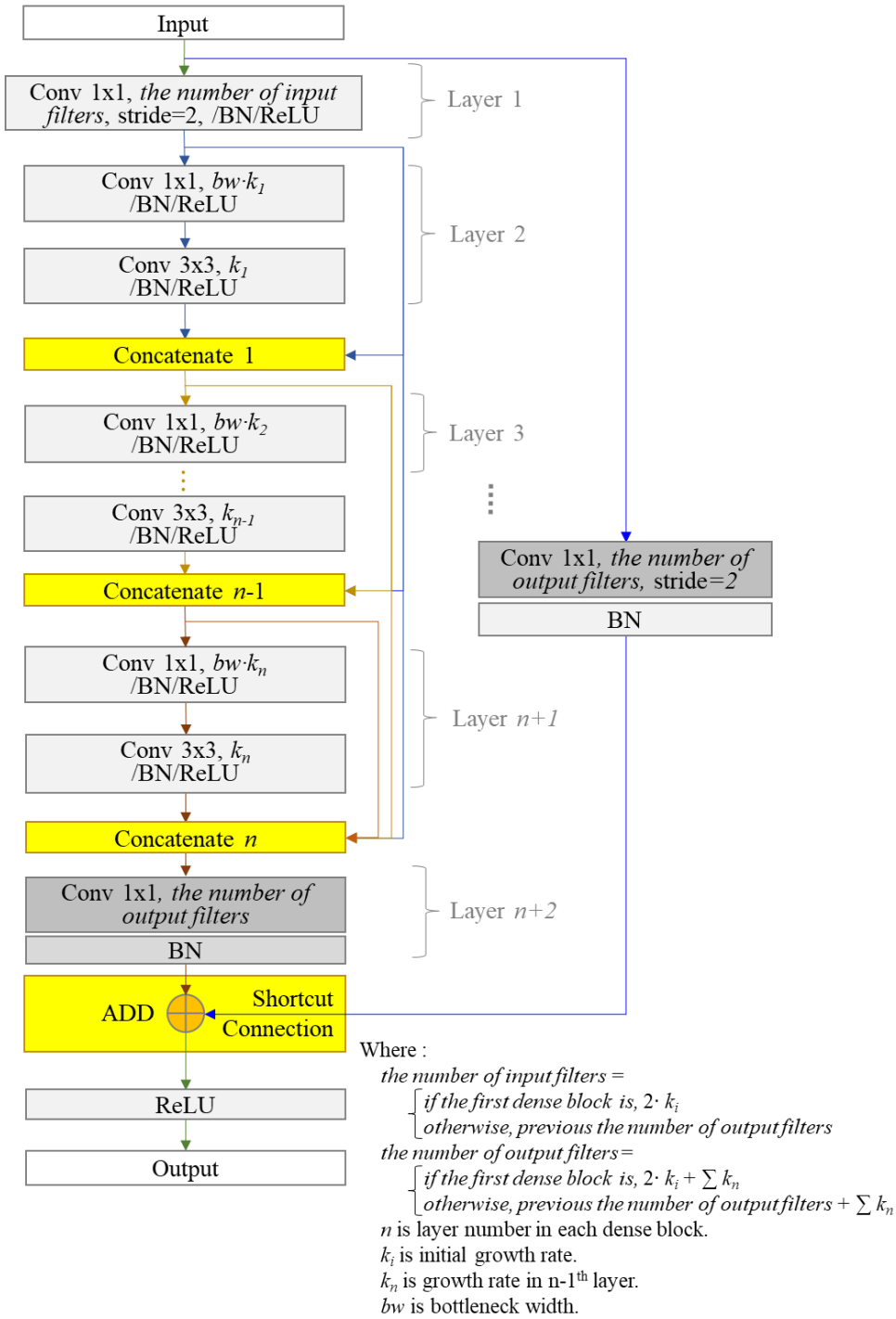


Figure 3.2 Dense block network model with post-activation in ISD.

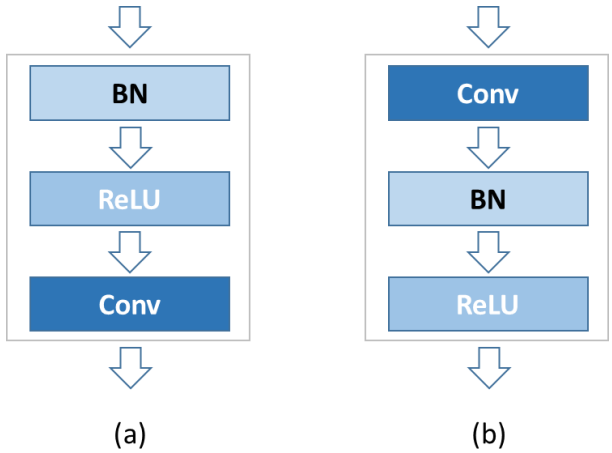


Figure 3.3 Comparison with pre-activation and post-activation. (a) Pre-activation of BN-ReLU-Conv (b) Post-activation of Conv-BN-ReLU in ISD.

However, crucially in contrast to DenseNet [91], ISD combine features through summation before they are passed into a dense block combined features by concatenating them with post-activation. Figure 3.2 illustrates this layout schematically. Santhanam et al. [135] presented the result that pre-activation ResNets consistently outperforms the original post-activation only at very high-network depths (≥ 152 depths). ISD has 38 or 42 depths at low-network depths and post-activation ISD outperformed pre-activation on the results of experiment. Figure 3.3 illustrates pre-activation and post-activation. Thus, in our approach, ISD has a structure with post-activation as shown in Figure 3.2.

Moreover, as illustrated in Figure 3.4, there is dynamic growth rate unlike DenseNet [91], which applies different growth rate in each layer in order to optimize the model. The growth rate that regulates the amount of information on each layer determine the

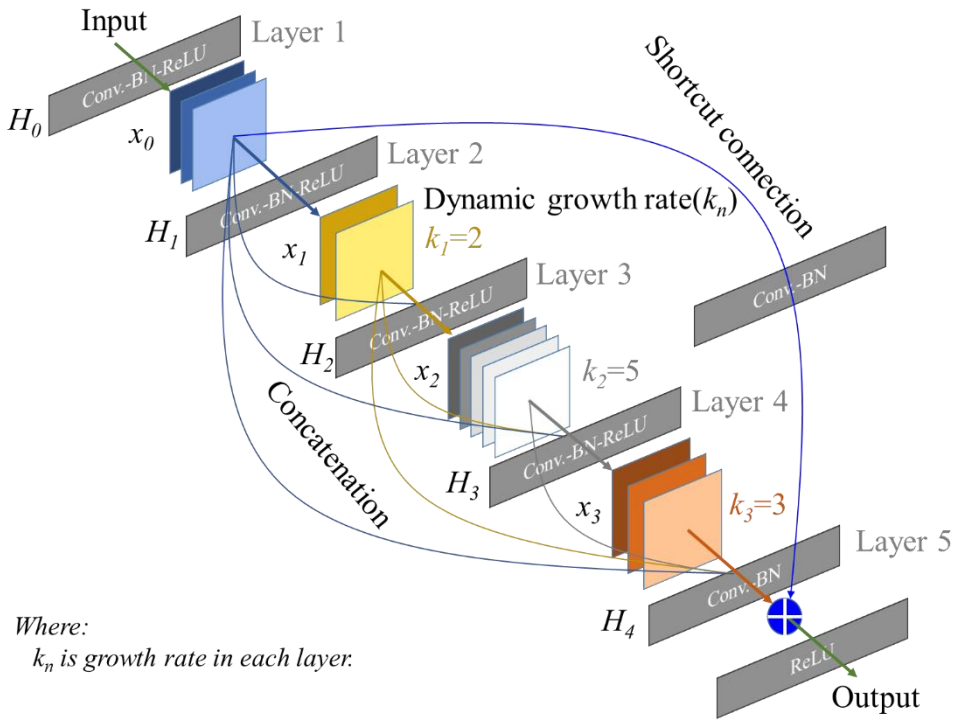


Figure 3.4 A dense block with dynamic growth rate of k = 2, 5, 3 in each layer on ISD.

number of feature-map. The dynamic growth rate substantially reduces the number of parameters, optimizing the model more compact and improving the performance. DenseNet has a structure that concatenates feature map according to the growth rate, thus the deeper layer is, the more multiple training parameters increase.

In order to reduce a huge number of training parameters due to concatenate, ISD can apply three dynamic growth rate methods. The first method increase or decrease the growth rate sequentially. There are three options as shown in Figure 3.5. The second is a method of compressing the generated feature map according to the growth rate. As shown in Figure 3.6, we use the sum module or the

mean module to fix the growth rate at one. The third is a method designed to improve performance, by concatenating the compressed feature map and the feature map generated by growth rate. As shown in Figure 3.7, the growth rate is always one more due to the compressed feature map concatenation. Details for the performance of three approaches will be given in the experiment section. Consequently, we design the ISD that can apply various types of dynamic growth rates.

Our architecture has the same concept of combining feature re-usage ability of ResNet [90] and feature re-exploration ability of DenseNet [91] as in DPN [99] and MixNet [100], However, as aforementioned in previous section 2.3, not only the structure, but also the followings are different. ISD improves DenseNet [91] by applying a dynamic growth rate to reduce the number of parameters and by using shortcut connection with same size in each block which is a group of layers to alleviate the gradient vanishing problem and to achieve superior efficiency with compactness. Furthermore, ISD performs down-sampling in a dense block rather than a transition layer and changes pre-activation to post-activation. Additionally, ISD removes pooling of transition layer to transfer more information and reduces channels in bottleneck layers for more compactness.

ISD has mainly two hyper-parameters: First, we refer to n as number of layers in each dense bock. Second, we refer to k as growth rate of the network. We optimized the hyper-parameters through experimental results.

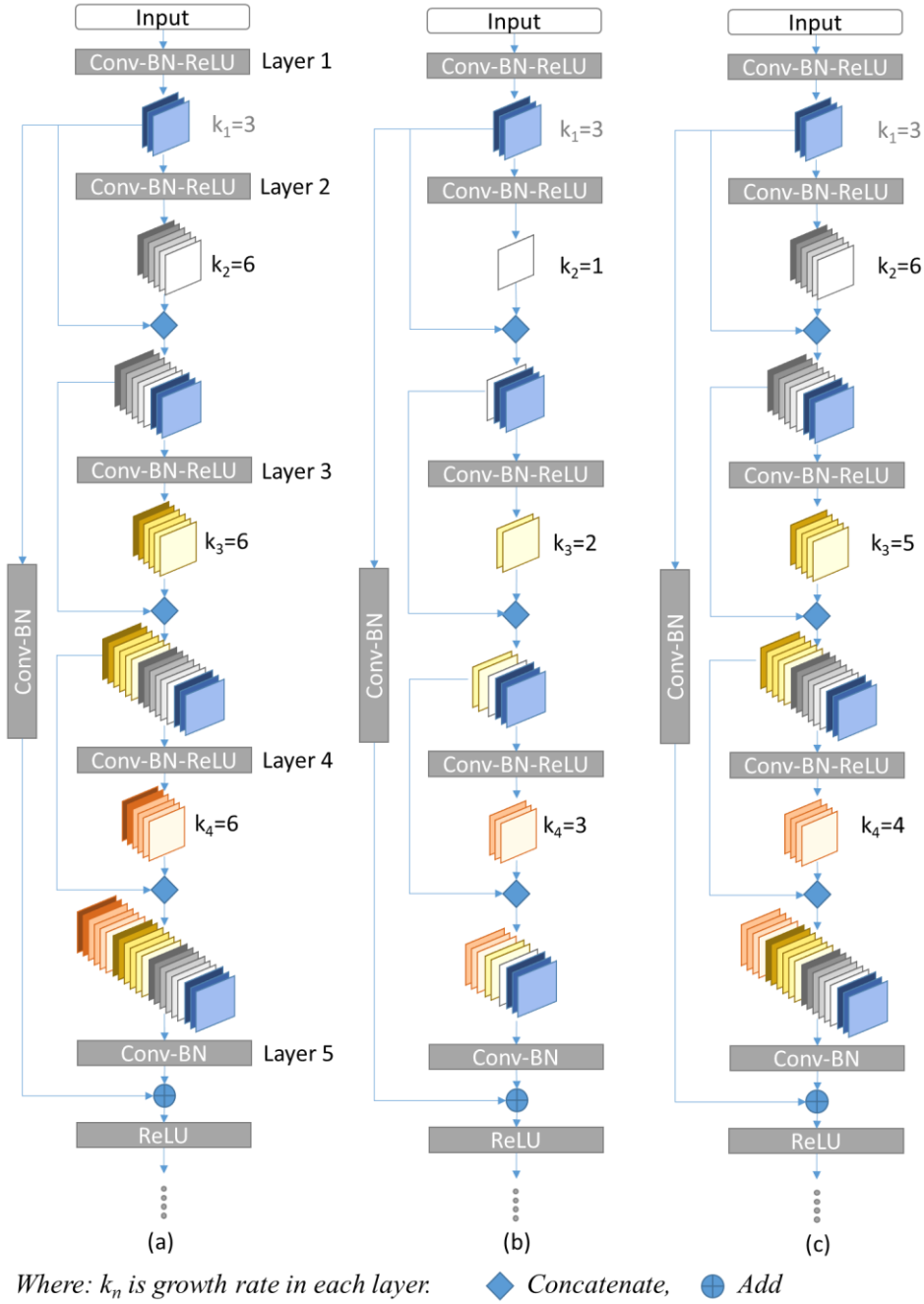


Figure 3.5 ISD structure with dynamic growth rate applying different growth rates in each layer. The dynamic growth rate is applied after the second layer. we compare three cases: (a) uniform growth rates (k=6,k,⋯,k) are used; (b) increasing growth rates (1,2,3,⋯,k=6) are used; (c) decreasing growth rates (k=6,k−1,k−2,⋯,2,1) are used.

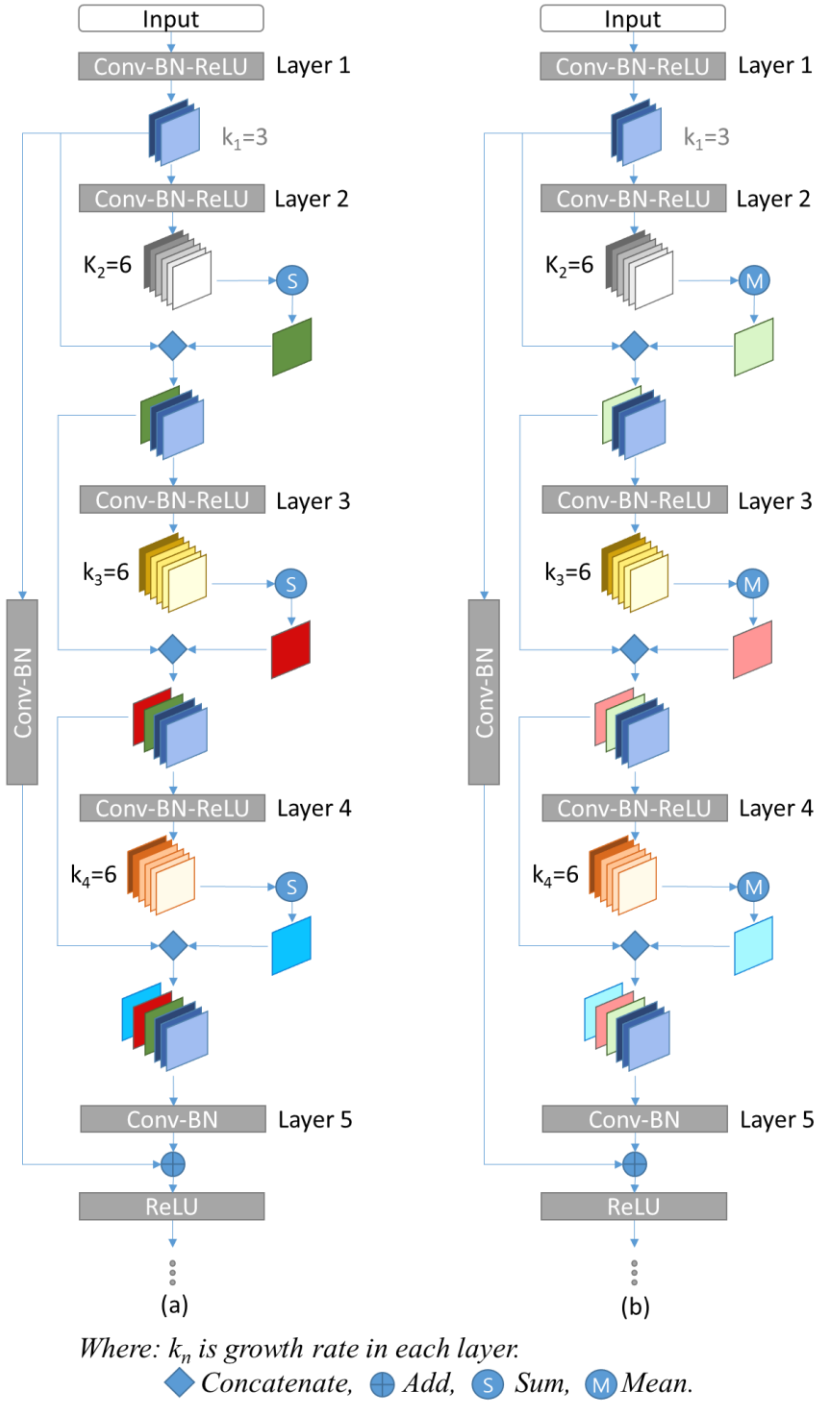


Figure 3.6 ISD structure to reduce training parameters increasing with growth rate. Regardless growth rate, it always grows by one. (a) Reduced by sum module with uniform growth rate; (b) Reduced by mean module with uniform growth rate.

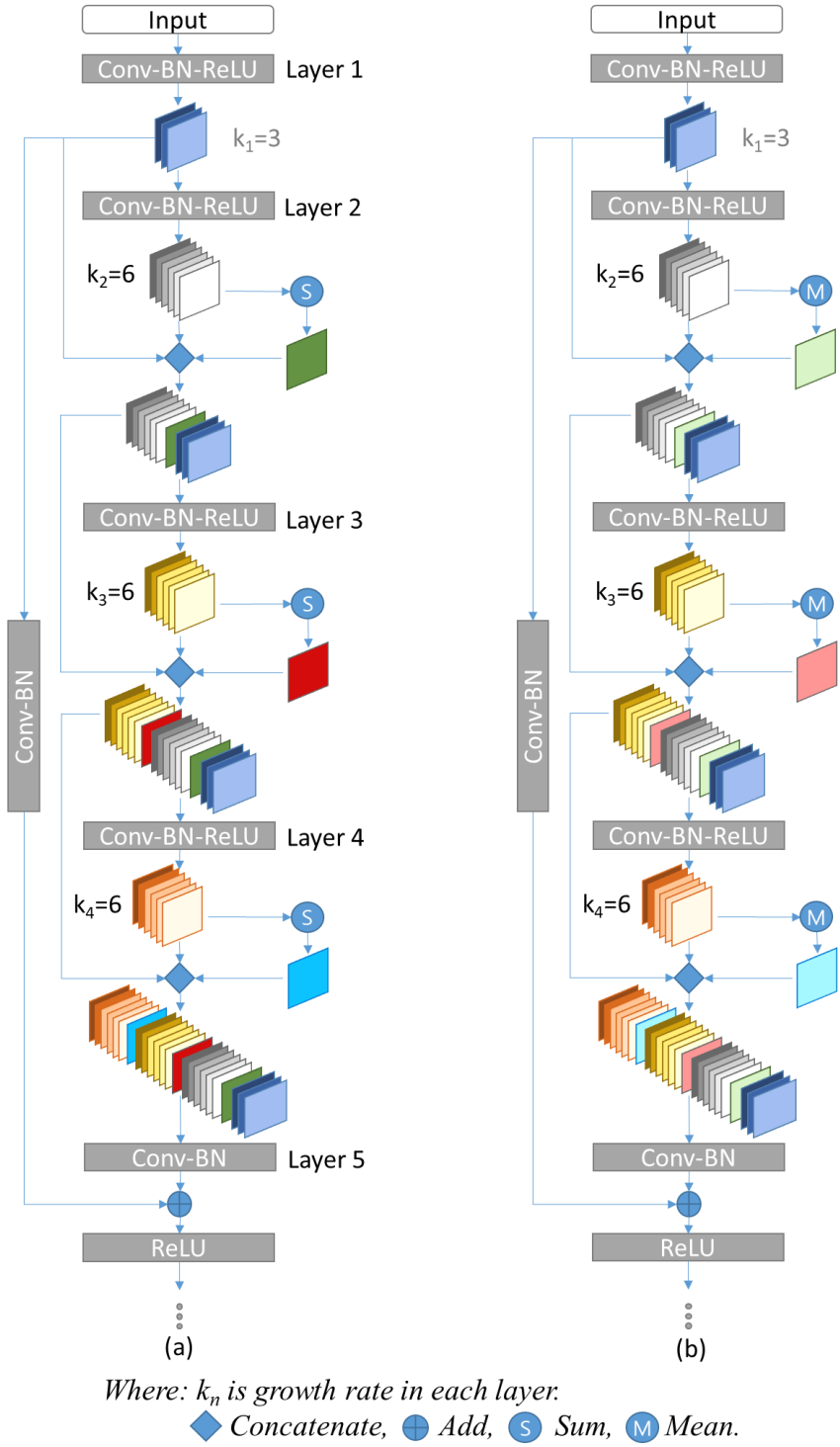


Figure 3.7 ISD structure in which the sum or mean module is added to the growth rate. (a) The sum module is added to uniform growth rate; (b) The sum module is added to uniform growth rate.

3.2. Pre-processing

Notably, computer vision system used in semiconductor photolithography process uses infrared and ultra-high-speed cameras with special manufactured lenses to acquire precise images. Therefore, the image is high quality. However, due to external factors such as various types of wafers, photoresists, the number of rotations of the motor, and diffuse reflection of light, the characteristics of the image are greatly changed or distortion occurs. Thus, rather than considering inputs for various image qualities, it is necessary to apply an image enhancement technique that extracts features by emphasizing contrast or boundary features for image segmentation.

Edge detection is one of the significant section of the image processing algorithms which have many applications like image morphing, pattern recognition, image segmentation and image extraction etc. As the edge is one of the major information contributors to any image, hence the edge detection is a very important step in many of the image processing algorithms. It represents the contour of the image which could be helpful to recognize the image as an object with its detected edges. Edge detection algorithms are fundamental importance for image

processing applications, because it's simply can determine within a short time the boundaries of objects in the image. Edge detection process as simple as can be explained in the following way; the intensity values of pixels which are neighbor each other are compared. During this process, the remarkable changes of density is called edge regions.

If image has noises, it should be well cleaned. Because noise affects the change of density in the image and it reduces the success rate for edge detection algorithms. To overcome noise problem, many studies were made for years and many different edge detection algorithms has emerged. With continuous development, edge detection algorithms have been used many areas thanks to the capability be able to simply use in short time and success rates increasing day to day.

Commonly used edge detection algorithms are Sobel, Roberts, Prewitt, Canny and LoG (Laplacian of Gaussian) edge detection algorithms which are still maintains its popularity today. Kabade et al. [136] proposed block level Canny edge detection algorithm which is the special algorithm to carry out the edge detection of an image in order to reduce the time and memory consumption.

In case of the suck-back state among the inspection types shown in Figure 1.4, it is hard to extract the feature from an image overlapped by nozzle image and photoresist image. In addition, the image of photoresist is varied by depending on the type of nozzle, and the image of nozzle is varied by depending on the kind of

photoresist. In practice, Ozturk et al. [137] compared Canny edge detection algorithm for their glass defect detection performance.

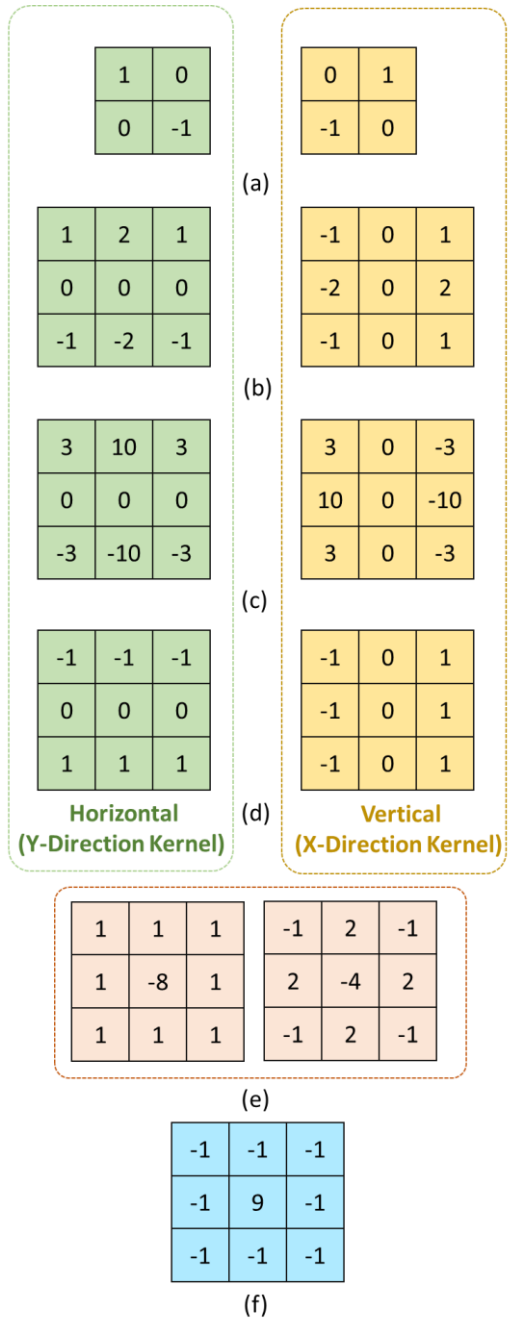


Figure 3.8 Convolution kernel for each operator. (a) Robert, (b) Sobel, (c) Scharr, (d) Prewitt, (e) LoG, (f) Sharpen.

If Canny edge detection algorithms applied to image with little noise or image that optimized image processing methods, they can get much better results.

As stated previously, the popular edge detection algorithms, such as Roberts, Scharr, Prewitt, Sobel, LoG, Sharpen, is shown in Figure 3.8. Robert edge detection operator has a fast and simple structure. It has 2×2 convolution kernels and these two convolution kernels is rotated 90° to each other. Sobel operator has two pieces and 3×3 kernels. These kernel maps is rotated 90° each other and are applied image with the convolution. Sobel operator is gradient based edge detection algorithms. Therefore, it use maximum points during the edge detection process. Scharr operator is similar to the method used by the Sobel operator. It is also divided into the x−direction and the y−direction. The difference is that the Scharr operator has a relatively large kernel value, so that the surrounding pixels will have a larger influence on the edge, and the edge will be more. Prewitt operator shows many similarities with the property of Sobel operator. It has two pieces kernels and these size is 3×3. It is gradient based edge detection operator and it has gradient features. Compared to the success of edge detection in complex image, success of Prewitt operator is greater than Roberts operator. Laplacian of an image reveals that fast changing points of density in the image. Because of this property, it can used edge detection. LoG filter take second derivative in the image and try to find zero−crossing points. Since the second derivative of the image is used, this filter is very sensitive

to noise. To overcome this problem, firstly the noise should be reduced by applying Gaussian smooth filter. And then Laplacian filter must be implemented to image. Laplacian pixel density value is calculated as shown in Equation (6).

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (6)$$

Commonly used 3×3 kernels to the LoG filter which is very successful in image processing applications that has less noise level. Additionally, Sharpen filter in image processing improves spatial resolution by enhancing the edges of objects and adjust the contrast and the shade characteristics: i) Highlight fine detail. ii) Enhance detail that has been blurred.

Figure 3.9 illustrates the improved image by implementing various edge detection algorithms. In order to inspect semiconductor photolithography process, the specific image filter modified by the sobel edge detector [138], which is composed of a pair of 3×3 convolution masks, one estimating gradient in the horizontal x−direction and the other estimating gradient in vertical y−direction, is adopt to identify points in an image at which the image brightness changes sharply or, more formally, has discontinuities.

Pre−processing is performed by using convolution on the image by means of the specific image filter. The edge occurs where there is a discontinuity in the intensity function or a very steep intensity

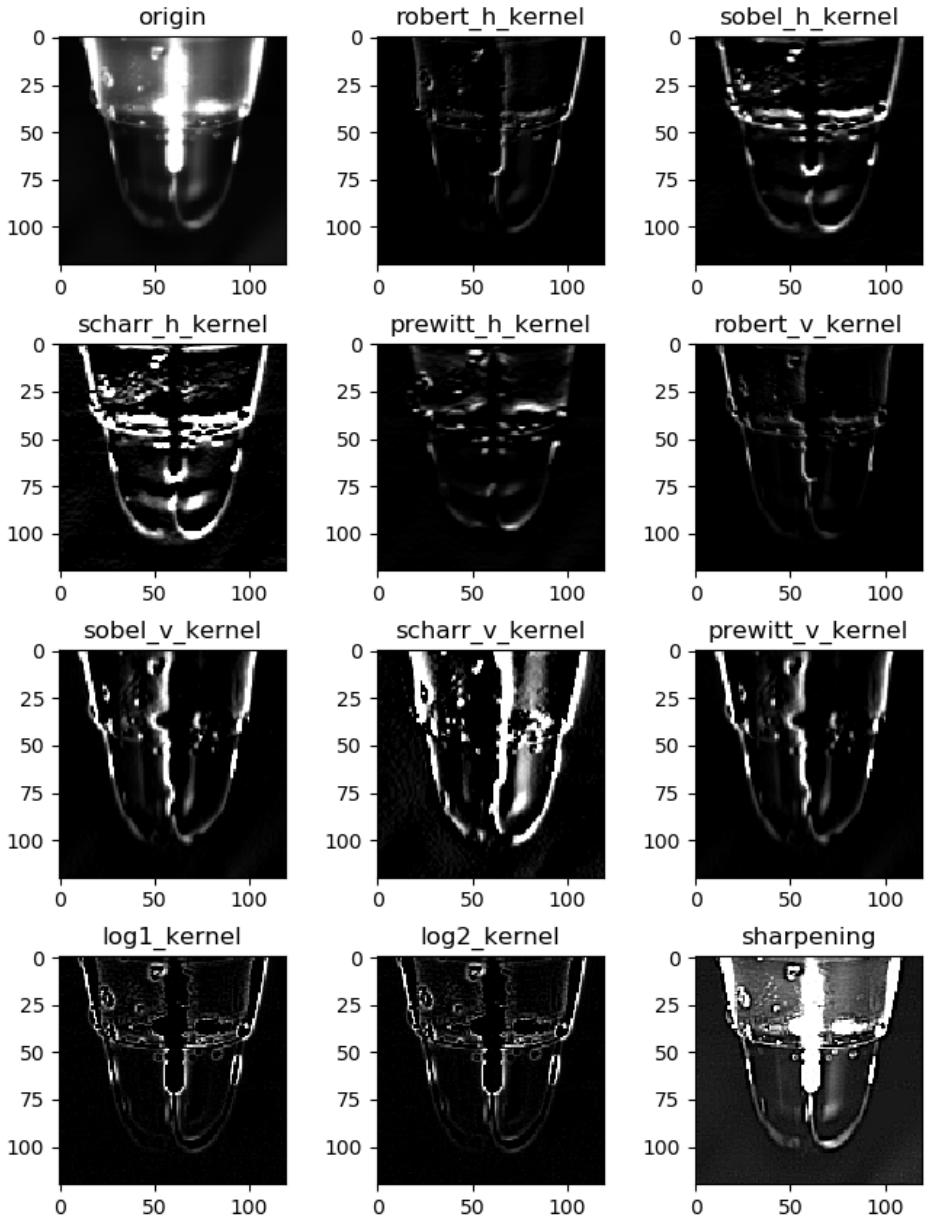


Figure 3.9 Implementation of edge detection algorithm to image. (a) Image is filtered by horizontal kernel (b) Image is filtered by vertical kernel.

gradient in the image. Thus, the edge could be located at which the derivative is maximum. The gradient is a vector, whose components measure how rapid pixel value are changing with distance in the x

and y direction. Thus, the components of the gradient may be found using the following approximation:

$$\frac{\partial f(x,y)}{\partial x} = \Delta x = \frac{f(x+dx,y) - f(x,y)}{dx} \quad (7)$$

$$\frac{\partial f(x,y)}{\partial y} = \Delta y = \frac{f(x,y+dy) - f(x,y)}{dy} \quad (8)$$

Where dx and dy measure distance along the x and y directions respectively. In discrete images, one can consider dx and dy in terms of numbers of pixel between two points, dx = dy = 1

$$\Delta x = f(x+1,y) - f(x,y) \quad (9)$$

$$\Delta y = f(x,y+1) - f(x,y) \quad (10)$$

The different operation in "(9)" and "(10)" correspond to convolving the image with the following image filter mask.

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -g & 0 & g \\ -1 & 0 & 1 \end{bmatrix} \quad (11)$$

$$\Delta y = \begin{bmatrix} -1 & -g & -1 \\ 0 & 0 & 0 \\ 1 & g & 1 \end{bmatrix} \quad (12)$$

In "(11)" and "(12)", g is adaptively applied according to the image intensity. The calculation formula is as follows:

$$g = \frac{1}{N_x \cdot N_y'} \sum_{x=0}^{N_x} \sum_{y=0}^{N_y'} \text{Image}(x, y) - \text{Image}(x, y + \text{offset}) \qquad (13)$$

Where:

N_x : the number of pixels in x coordinate

N_y : the number of pixels in y coordinate

N_y' : N_y − offset

The image pre-processed by means of the specific image filter is shown in Figure 3.10. The pre-processed image that is used as the input of ISD has significance in extracting the enhanced feature-map for inspection.

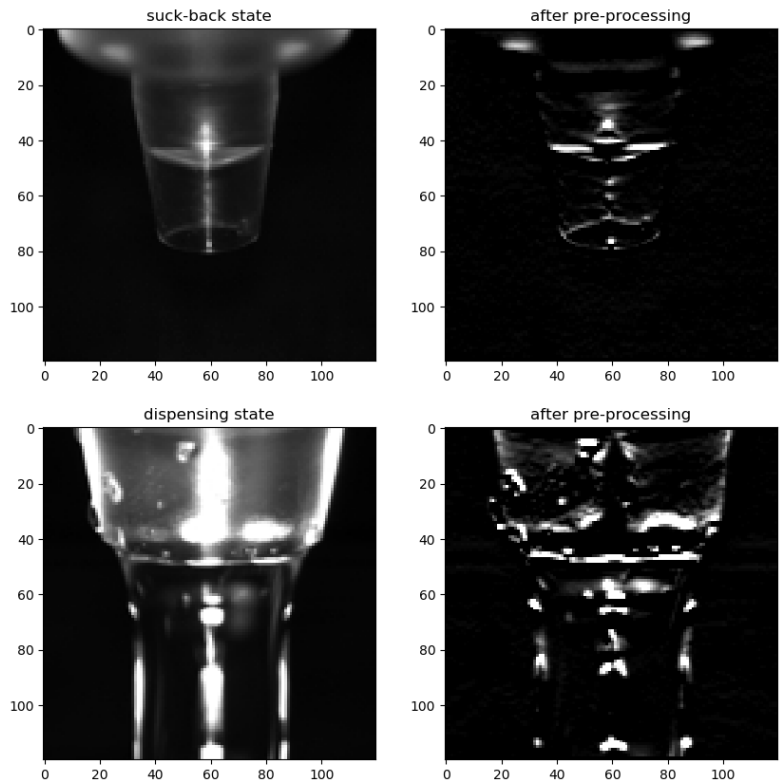


Figure 3.10 The image pre-processed by means of our filter of equation "(12)" (g = 4).

3.3. Model Training

In our approach, we focus on the image segmentation task without using the pre-trained models. We train models on target dataset directly without using IamgeNet dataset as shown in Figure 3.11. Thus, our approach which is learning deep models from scratch has very appealing advantages over existing solutions. ISD is trained with various nozzle image as shown in Figure 3.11, to classify the nozzle type. Among ISD models trained up to 100 epochs, the weight of ISD model with the best performance is used for training Mask R-CNN for image segmentation. The image dataset used to train Mask R-CNN is prepared by using image annotation tool (i.e. VGG image annotator) which manipulates the labeled segmentation of image. In addition, filtering the input dataset is performed for pre-processing of training model.

Figure 3.12 illustrates the training process. Training performs in two stages. The first is training for classification and the second is training for image segmentation. The weights of model obtained from training for classification, are used in training for image segmentation. Training for image segmentation performs with labeled and pre-processed images. After that, the weights of model obtained from training for image segmentation, are used in inference model.

Inference model for evaluation uses only original image without pre-processing.

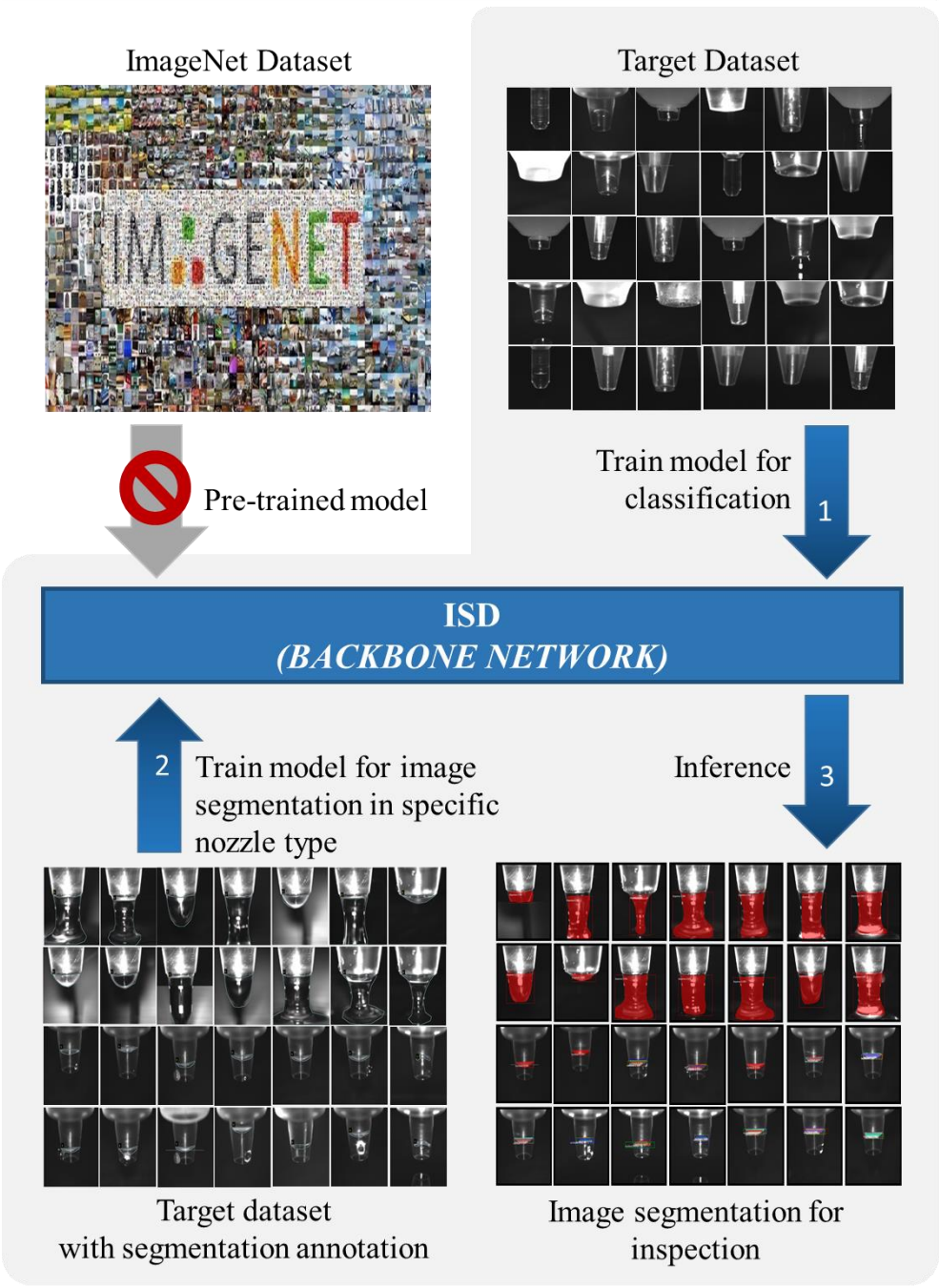


Figure 3.11 Illustration of training model on target dataset directly.

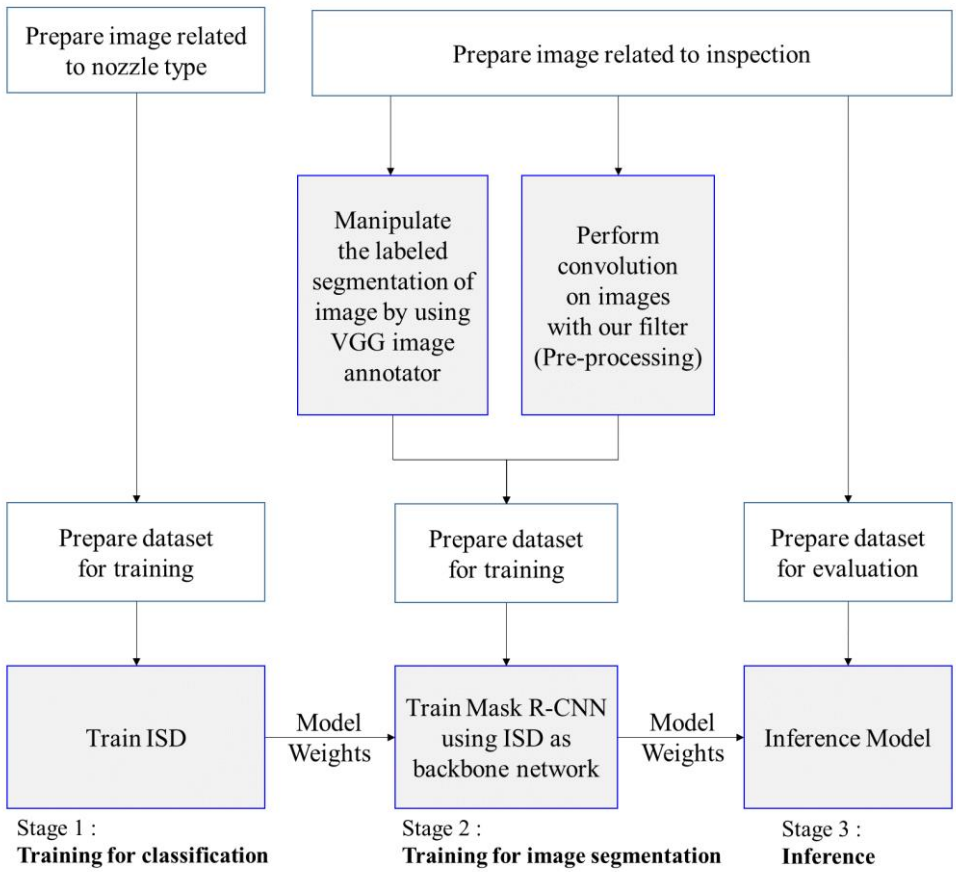


Figure 3.12 Training process of image segmentation using ISD as the backbone network of Mask R-CNN framework.

3.4. Training Objective

The training objective is the losses being used to converge the huge number of weights and the hyper-parameters that must be conducive

to this convergence.

In training model for classifying nozzle type, categorical cross entropy loss generally used to classify image is adopt to the loss of ISD (i.e. L_{ISD}). It is a softmax activation plus a cross entropy loss.

$$L_{ISD} = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right) \tag{14}$$

Where:

s_p = the CNN score for the positive class

C = the number of classes

s_j = the score inferred by the network for each class in C

In training model for detecting nozzle state, the training loss is adopt from Faster R-CNN and Mask R-CNN, which is a weighted sum of the classification loss (cls), the localization loss (box) and segmentation mask loss (mask). Where L_{total_cls} and L_{total_box} are same as in Faster R-CNN [7] and L_{total_mask} is same as in Mask R-CNN [8].

$$L_{total} = L_{total\_cls} + L_{total\_box} + L_{total\_mask} \tag{15}$$

$$L_{total\_cls} = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) \tag{16}$$

$$L_{cls}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log (1 - p_i) \tag{17}$$

Where:

p_i = Predicted probability of anchor i being an object

p_i^* = Ground truth label of whether anchor i is an object

N_{cls} = Normalization term, set to be batch size

$$L_{total\_box} = \frac{\alpha}{N_{box}} \sum_i p_i^* \cdot L_1^{smooth}(t_i - t_i^*) \quad (18)$$

Where:

t_i = Predicted four parameterized coordinates

t_i^* = Ground truth coordinates

N_{box} = Normalization term, set to the number of anchor locations

α = Balancing parameter

$$L_{total\_mask} = -\frac{1}{m^2} \sum_{1 \le i,j \le m} \left[y_{ij} \log \hat{y}_{ij}^k + (1 - y_{ij}) \log (1 - \hat{y}_{ij}^k) \right] \quad (19)$$

Where:

y_{ij} = Label of cell(i,j) in the true mask for the region of size $m \times m$

$\hat{y}_{ij}^k$ = Predicted value of the same cell for the ground truth class k

3.5. Setting and Configurations

We adopt L2 regularization technique [139] to prevent the emergence of network training time "over-fitting" and eliminate the need for dropout. Furthermore, we adopt real-time data

augmentation to prevent overfitting on training with limited data in training model for classifying nozzle type. In order to obtain the higher performance, we set compression factor(θ) in transition layer to 1. Compression factor $\theta = 1$ means that there is no feature map reduction in the transition layer on dense blocks. Furthermore, we set the number of channels in bottleneck layer to $1 \cdot k$ (growth rate). All conv-layers are initialized with the "Kaiming He Initialization" method [140]. We have our own learning rate scheduling and mini-batch size settings. Details will be given in the experiment section. Configurations used in training model for detecting nozzle state, is illustrated in Table 3.2.

Table 3.2 Configurations used in training model for image segmentation.

Configuration	Value
The strides of each layer of the FPN Pyramid	[4, 8, 16, 32, 64]
Size of the fully-connected layers in the classification	8
Size of the top-down layers used to build the feature pyramid	256
Non-max suppression threshold to filter RPN proposals	0.9
How many anchors per image to use for RPN training	512
Mini-mask Shape (Height, Width)	[28, 28]
Input image resizing (Min, Max)	[32, 128]
Mean pixel	[123.7, 116.8, 103.9]
Number of ROIs per image to feed to classifier/mask heads	128

Configuration	Value
Percent of positive ROIs used to train classifier/mask heads	0.33
Shape of output mask	[28, 28]
Maximum number of ground truth instances to use in one image	100
Minimum probability value to accept a detected instance	0.9
Non-maximum suppression threshold for detection	0.7
Learning Rate	0.01
Learning Momentum	0.9
Weight decay regularization	0.0001

Chapter 4. Experimental Evaluation

We implement ISD based on the tensorflow platform [141]. The hardware platform is notebook with two GPUs as illustrated in Table 4.1. All our models are trained from scratch on NVidia GeForce GTX GPU.

Due to image related to semiconductor process is not available in open datasets for deep learning such as ImageNet, MS COCO, pascal VOC etc., the experimental dataset is acquired from computer vision system embedded in the currently operating semiconductor manufacturing equipment for photolithography inspection.

Table 4.1 Hardware specification.

Item	Specification
CPU	Intel Core i7-8750H 2.2GHz
Memory	16GB, 3200MHz DDR4
GPU0	Intel UHD Graphics 630
GPU1	NVIDIA GeForce GTX 1050 Ti

The size of image is 640×495 pixels and gray color. Intuitively, larger input images will bring better performance for image segmentation. However, an additional difficulty is that real world applications like computer vision system demand inspection to be solved in real-time. Fastest detectors are usually better than the best performing ones. Thus, we reduced the size of image used as the input of ISD to 120×120 pixels.

As aforementioned in previous section 3.3, we perform training model in two stages. The first is to train model for classifying nozzle type. The second is to train model for image segmentation in each suck-back state and dispensing state. In order to classify nozzle type, 18,304 images that have already been correctly classified into 8 types of nozzle, were collected from currently operating semiconductor manufacturing equipment. Then, we reduce the size of 18,304 images from 640×495 pixels to 120×120 pixels. We split these images randomly into 13,728 training datasets and 4,576 validation datasets at a ratio of 7:3. Then, we use these images to train model for classifying 8 nozzle types. In order to train model for image segmentation of suck-back state in one type of nozzle, we use a total of 410 images of one type of nozzle with 120×120 pixels. We split these images randomly into 266 training datasets and 144 validation datasets at a ratio of 6:4. Then, we use images manipulated with the labeled segmentation by using image annotation tool (i.e. VGG image annotator) to train model for image segmentation of suck-back state in specific nozzle type. In order to train model for

image segmentation of dispensing state in one type of nozzle, we use a total of 501 images of one type of nozzle with 120×120 pixels. We split these images randomly into 351 training datasets and 150 validation datasets at a ratio of 7:3. Then, we use images manipulated with the labeled segmentation by using image annotation tool to train model for image segmentation of dispensing state in specific nozzle type. Notably, in image segmentation of suck-back state, where the image segmentation area is much small, we increase the dataset for validation to prevent overfitting. Furthermore, as illustrated in Figure 4.1, we experimentally verify that our model is not overfitting. We evaluate performance with a new test dataset that has never been used for training, and compare results. We use new 194 images to verify overfitting in image segmentation of suck-back state. We use new 210 images to verify overfitting in image segmentation of dispensing state.

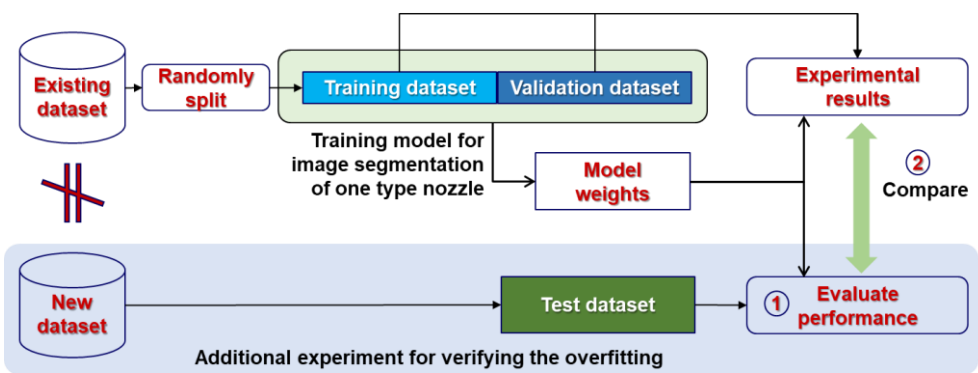


Figure 4.1 The method for experimentally verifying that our model is not overfitting.

We evaluate ISD with different depth and growth rates for compactness. We verify the effectiveness of the method through the comparison experiment. A consistent setting is imposed on all the experiments, unless when some components or structures are examined. As stated previously, we adopt the standard mean Average Precision (mAP) to measure the image segmentation performance.

4.1. Classification Results on ISD

The initial learning rate is set to 0.1 until 50th epoch and then divided by 10 after every 25 epochs. Our model is trained up to 100 epochs. The number of training steps per epoch is 429. The classification training accuracy after only 12 epoch is 99.71% and the validation accuracy is 99.69% for classifying nozzle type. The classification training and validation accuracy in each epoch is illustrated in Figure 4.2. The classification training and validation loss with shortcut connection on ISD model is illustrated in Figure 4.3, and The classification training and validation loss without shortcut connection on ISD model is illustrated in Figure 4.4. In addition to classification of nozzle type, we also test to detect image segmentation of nozzle type. We used 385 training dataset and 138 validation dataset for image segmentation. Figure 4.5 illustrates the result on detecting image segmentation in each nozzle type.

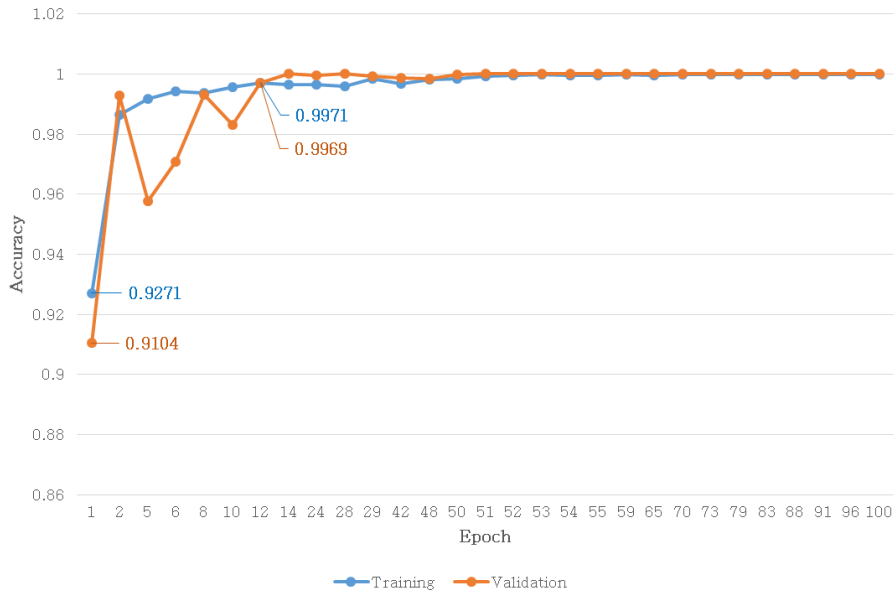


Figure 4.2 The classification training and validation accuracy in each epoch. ISD has 42 depths with shortcut connection. The uniform growth rate (k) is 6.

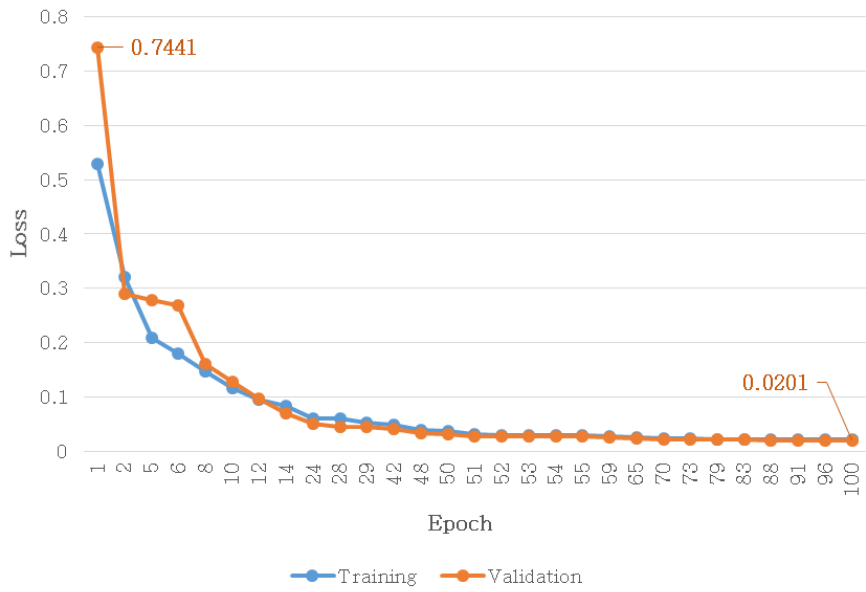


Figure 4.3 The classification training and validation loss in each epoch. ISD has 42 depths with shortcut connection. The uniform growth rate (k) is 6. The average processing time for each epoch is 31 seconds.

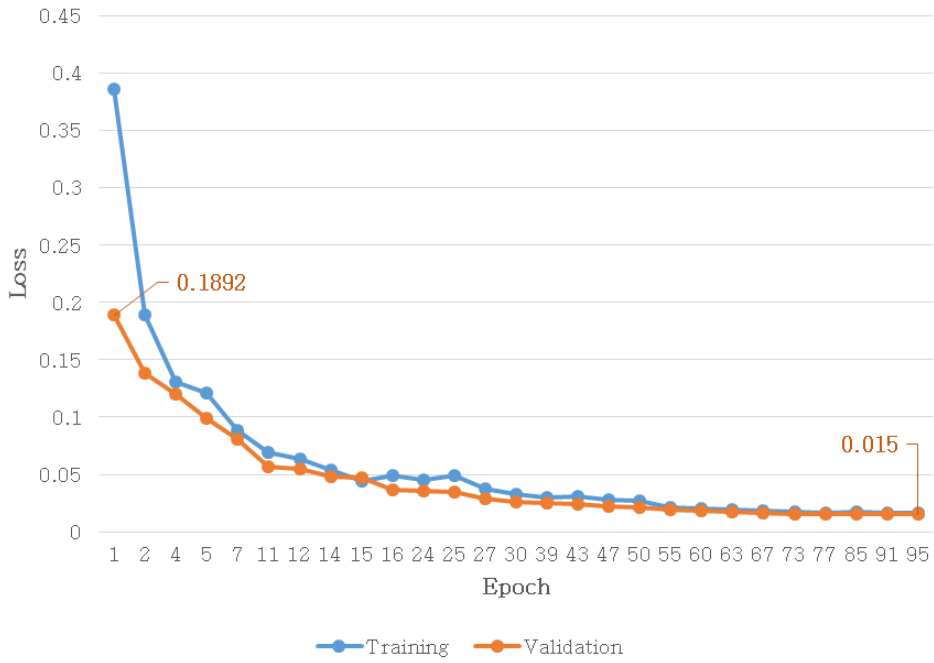


Figure 4.4 The classification training and validation loss in each epoch. ISD has 42 depths without shortcut connection. The uniform growth rate (k) is 6. The average processing time for each epoch is 26 seconds.

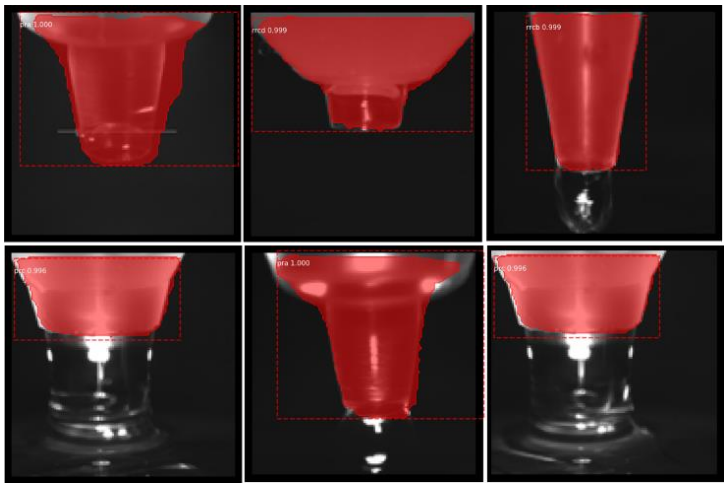


Figure 4.5 The experimental result of detecting image segmentation of nozzle type.

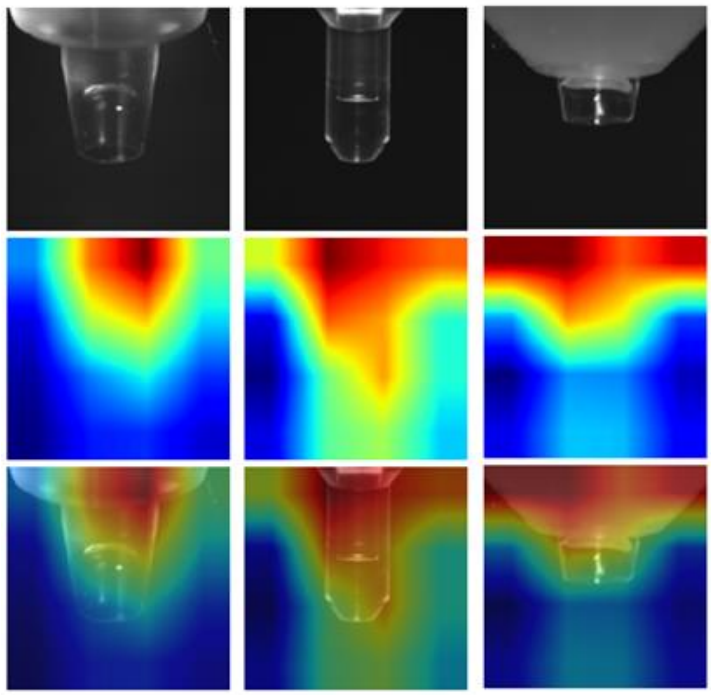


Figure 4.6 Visualization of class activation mapping, using ISD as backbone networks.

As illustrated in Figure 4.6, we visualize the class activation mapping (CAM) using GradCAM [142], which is commonly used to localize the discriminative regions for image classification. To optimize the ISD for classifying nozzle types, we conduct with various depths and shortcut connections. As shown in Table 4.2, comparing the validation loss according to various depths and shortcut connection, we can observe that the loss tends to decrease as the depth increases, while the difference is little. Besides, while the loss is greater in the case of having shortcut connection, interestingly observing the results of the image segmentation, it can be seen that the case of having shortcut connection has better performance.

Table 4.2 Comparison of loss according to various depths and shortcut connection in nozzle type classification results. The loss represents the classification training and validation loss, when the lowest loss in validation is obtained during the training up to 100 epoch. The uniform growth rate (k) is 6.

	ISD		Loss		Time
Depth	**Shortcut**	**Parameters**	**Training**	**Validation**	**Seconds**
24		15,038	0.0238	0.0211	19
	√	26,666	0.0303	0.0281	22
34		33,152	0.0211	0.0196	23
	√	60,560	0.0258	0.0241	29
46		57,272	0.0180	0.0163	28
	√	105,632	0.0239	0.0222	33
54		80,000	0.0165	0.0148	30
	√	148,832	0.0229	0.0211	38
62		106,472	0.0161	0.0145	35
	√	199,376	0.0222	0.0205	45

4.2. Comparison with Pre-processing

Our model for image segmentation is trained on the basis of two separate data sets to detect suck-back state and dispensing state. Our model is trained up to 100 epochs. The number of training steps per epoch is 100. The performance is evaluated by the model weight

with the lowest loss obtained during the training up to 100 epochs.

Comparison of suck-back state results. We set the learning rate to 0.01. The average processing time for each epoch is 209 seconds. The image segmentation training and validation loss in each epoch, is illustrated in Figure 4.7. We evaluate the performance of pre-processing on image segmentation task in the standard mean average precision. In aspect of the mask, the mask of nozzle type was detected well even without pre-processing using image filter.

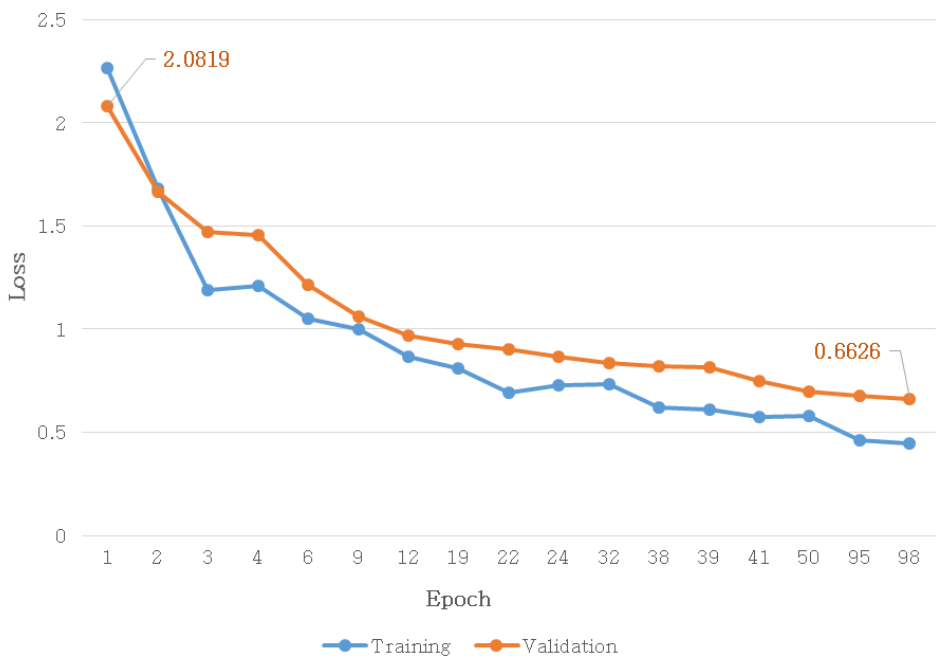


Figure 4.7 The image segmentation training and validation loss in each epoch on suck-back state results. ISD has 38 depths with shortcut connection. The uniform growth rate (k) is 6.

However, the mask of suck-back state for inspection is not detected or incorrectly recognized when the pre-processing is not performed. Figure 4.8 illustrates comparison with pre-processing in aspect of mask. We can observe that the pre-processing using image filter can achieve higher accuracy, which is consistent to our conjecture that the enhanced feature-map is extracted by pre-processing. In aspect of performance, comparison of pre-processing is illustrated in Table 4.3. We explore the effect of pre-processing. Our approach is simple and highly effective. mAP@0.50 in validation is improved by 4.27% when the pre-processing is performed.

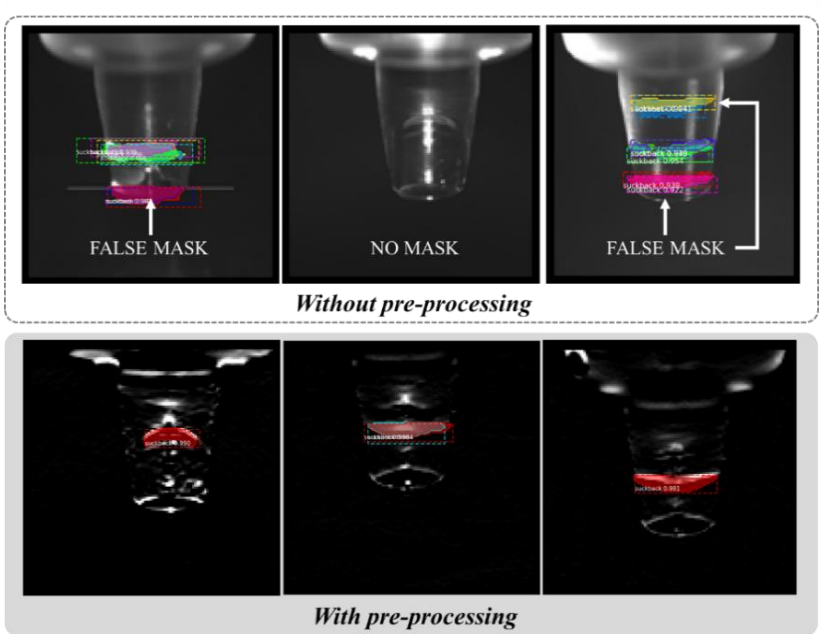


Figure 4.8 Image segmentation of suck-back state for inspection. In case of training ISD with pre-processing, the mask performance is better than without pre-processing. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

Table 4.3 Comparison of performing pre-processing in suck-back state results. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

ISD	Test (mAP, %)		Train (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75	IoU[1]:0.50	IoU[1]:0.75
w/o pre-processing	90.97	38.95	87.97	43.87
w/ pre-processing	95.24	57.14	95.42	68.67

[1] Intersection over Union.

Table 4.4 Comparison of performance according to pre-processing algorithm in suck-back state results. ISD has 42 depths with shortcut connection and the uniform growth rate (k) is 6.

Pre-processing Algorithm	Test (mAP, %)		Train (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75	IoU[1]:0.50	IoU[1]:0.75
Our Filter	95.49	59.42	97.74	65.21
Robert	86.81	50.69	87.22	62.86
Sobel	91.67	45.49	88.91	50.80
Scharr	91.32	53.88	87.78	58.35
Prewitt	89.58	47.51	92.29	55.91
LoG	93.75	53.73	95.87	62.66
Sharpen	92.01	52.26	94.17	57.96
CLAHE	85.21	45.26	86.17	55.01

[1] Intersection over Union.

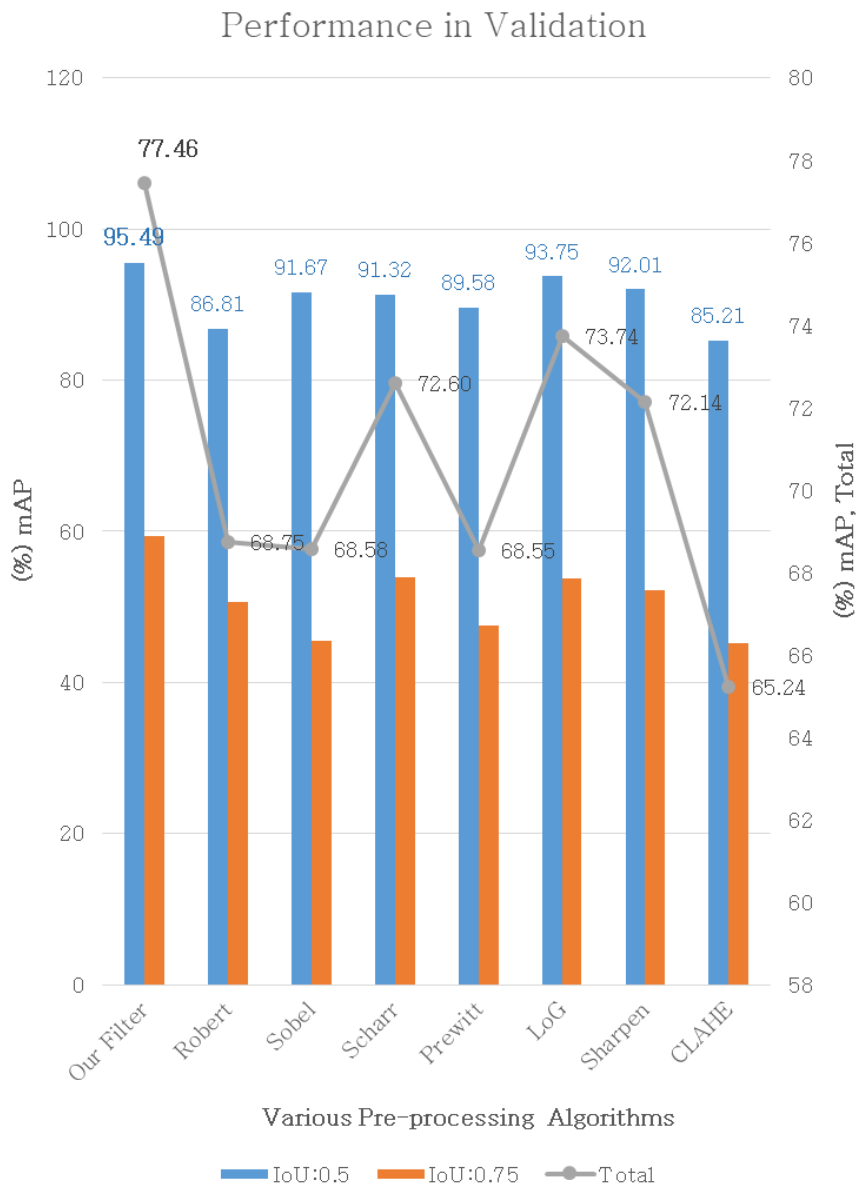


Figure 4.9 The chart shows performance in validation for comparison of various pre-processing algorithms in suck-back state results. ISD has 42 depths with shortcut connection and the uniform growth rate (k) is 6

Interestingly, mAP@0.75 in validation is improved with a large margin (18.19%) when the pre-processing is performed. We can observe that the greatest task performance improvement was yielded

by pre-processing. We was able to achieve remarkable improvement. Furthermore, comparison of performance according to various pre-processing algorithm is illustrated in Table 4.4. We observe that using our filter is much better performance with a large margin (10.28%) than using CLAHE algorithm, at mAP@0.50 in validation. As illustrated in Figure 4.9, we empirically demonstrate that pre-processing using our designed filter has better performance than other algorithms.

Comparison of dispensing state results. We set the learning rate to 0.01. The average processing time for each epoch is 55 seconds. The image segmentation training and validation loss in each epoch, is illustrated in Figure 4.10.

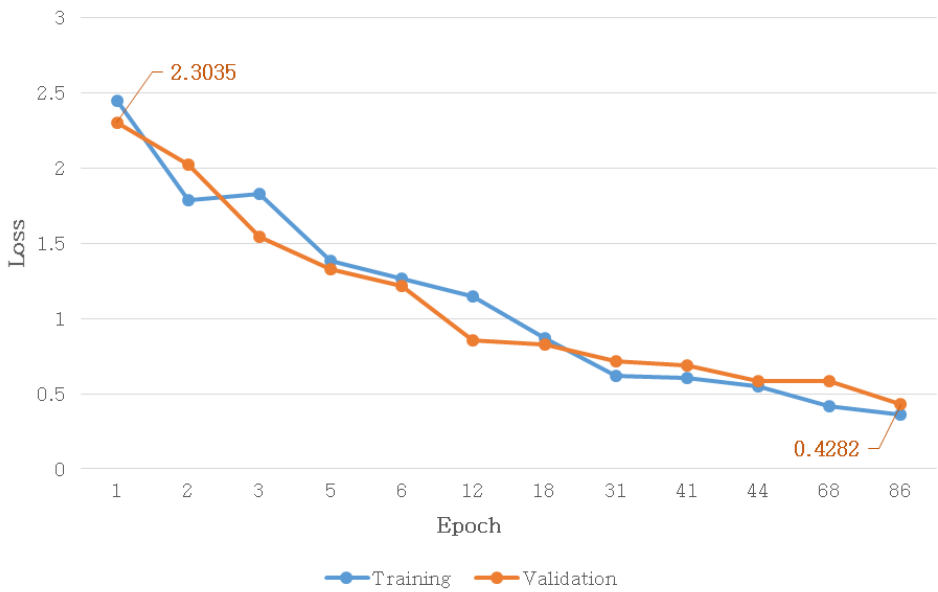


Figure 4.10 The image segmentation training and validation loss in each epoch, on dispensing state results. ISD has 38 depths with shortcut connection. The uniform growth rate (k) is 6.

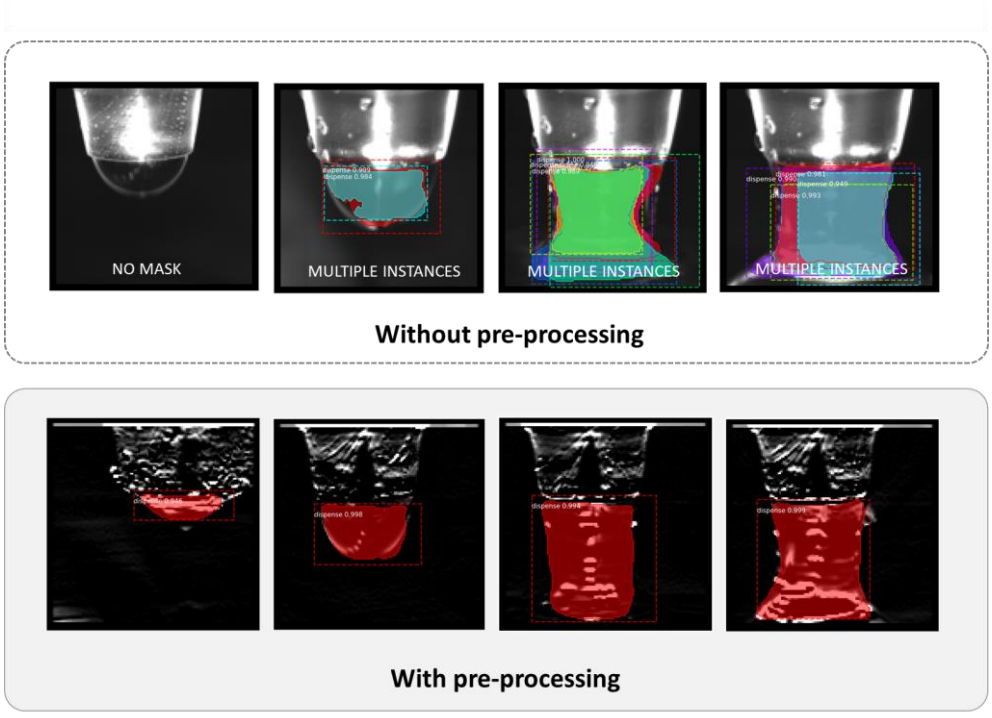


Figure 4.11 Image segmentation of dispensing state for inspection. In case of training ISD with pre-processing, the mask performance is better than without pre-processing. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

We evaluate the performance of pre-processing on image segmentation task in the standard mean average precision as well as suck-back state results. As illustrated in Figure 4.11, the mask of dispensing state for inspection is detected as multiple images or incorrectly recognized when the pre-processing is not performed. We can observe that the pre-processing using our filter can achieve higher accuracy as well.

In aspect of performance, comparison of pre-processing is illustrated in Table 4.5. mAP@0.50 in validation is improved by 2.33%

when the pre-processing is performed. Additionally, mAP@0.75 in validation is improved with more margin (5.5%) when the pre-processing is performed. We can observe that the greatest task performance improvement was yielded by pre-processing even in dispensing state as in suck-back state.

Table 4.5 Comparison of performing pre-processing in dispensing state results. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

ISD	Test (mAP, %)		Train (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75	IoU[1]:0.50	IoU[1]:0.75
w/o pre-processing	96.00	72.17	96.87	73.43
w/ pre-processing	98.33	77.67	96.30	76.21

[1] Intersection over Union.

Table 4.6 Comparison of performance according to pre-processing algorithm in dispensing state results. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

Pre-processing Algorithm	Test (mAP, %)		Train (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75	IoU[1]:0.50	IoU[1]:0.75
Our Filter	98.33	77.67	96.30	76.21
Robert	84.33	72.33	90.74	69.59
Sobel	95.33	72.39	96.30	76.45
Scharr	97.00	78.39	97.15	78.63

Pre-processing Algorithm	Test (mAP, %)		Train (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75	IoU[1]:0.50	IoU[1]:0.75
Prewitt	96.36	74.19	95.51	78.97
LoG	95.33	77.67	96.01	75.74
Sharpen	96.00	69.89	96.01	76.64
CLAHE	95.33	75.72	96.96	73.80

[1] Intersection over Union.

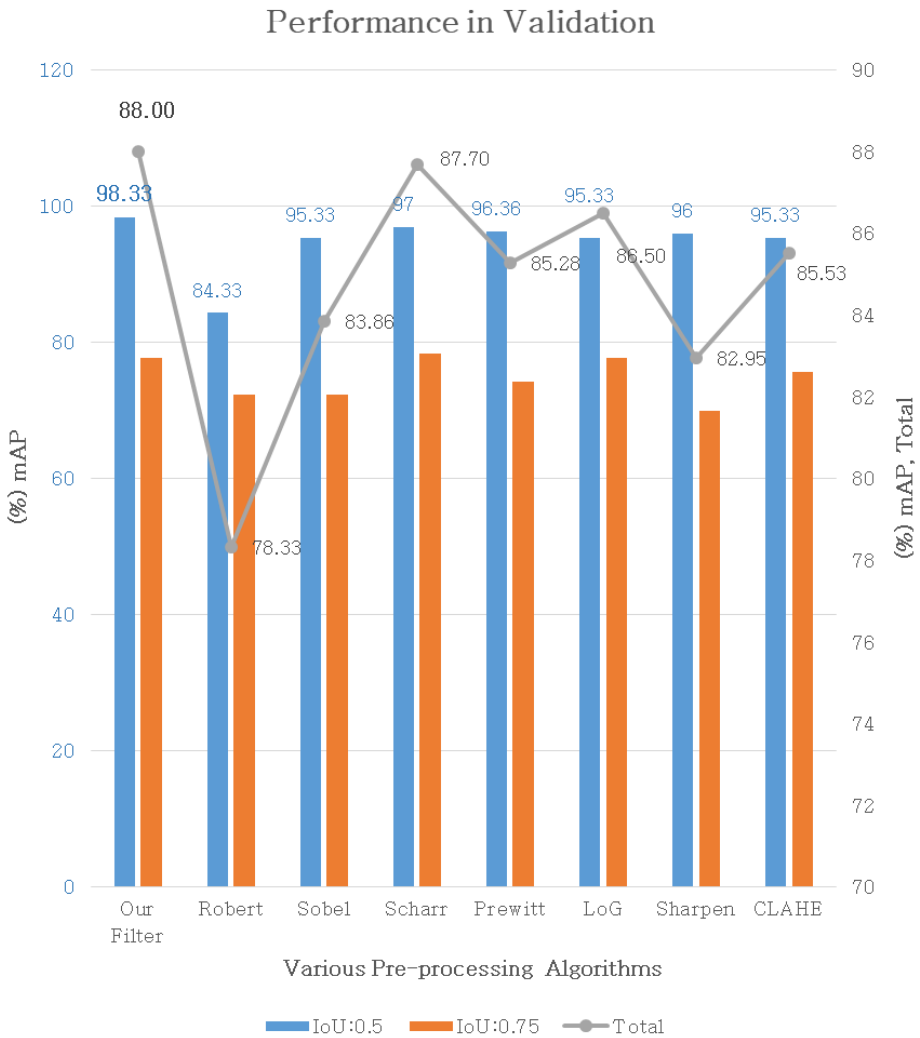


Figure 4.12 The chart shows performance in validation for comparison of various pre-processing algorithms in dispensing state results. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6.

Furthermore, comparison of performance according to pre-processing algorithm is illustrated in Table 4.6. We observe that using our filter is much better performance with a large margin (14%) than using Robert algorithm, at mAP@0.50 in validation. As illustrated in Figure 4.12, the experimental results show that pre-processing using our designed filter has consistently better performance than other algorithm, as in suck-back state result.

4.3. Image Segmentation Results on ISD

Model optimization and performance are an important trade-off for the applications of deep neural networks in image segmentation tasks for real-time application. In order to optimize ISD, we conduct experiments with three cases which are the number of depth, shortcut connection and dynamic growth rate. Our experiments are conducted on object categories which are suck-back state and dispensing state respectively.

4.3.1. Results on Suck-back State

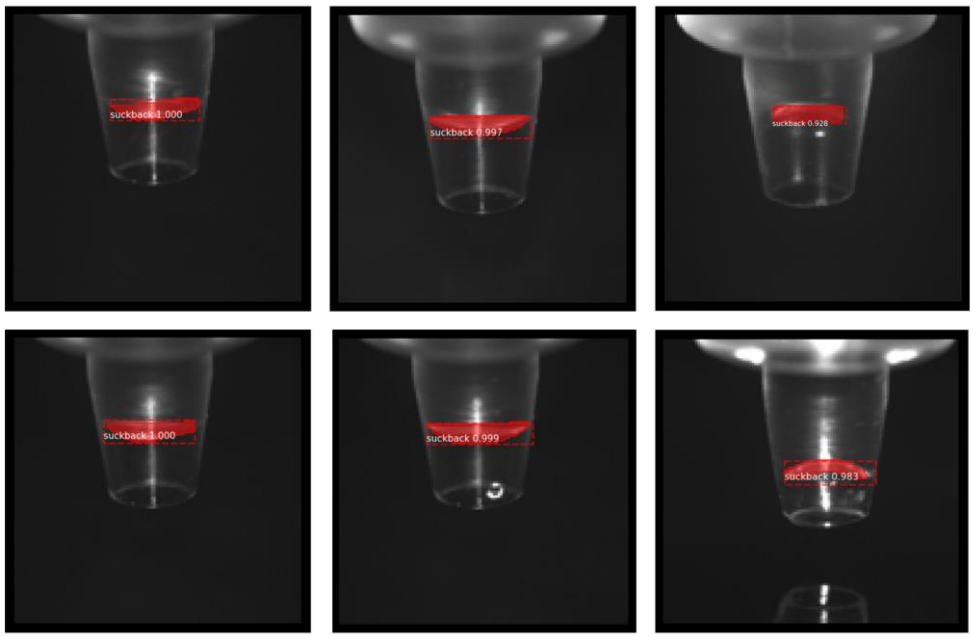


Figure 4.13 An example of image segmentation results on suck-back state.

In this section, in order to optimize our model with more compactness for detecting suck-back state, we justify the effectiveness of each design structure and parameters elaborated earlier. An example of image segmentation results on suck-back state is shown in Figure 4.13.

The number of depth. We have experimented with various depths on ISD for detecting suck-back state. The main results are summarized in Table 4.7. We conjecture that the deeper layer is the better performance, as is well known. However, as shown in Figure 4.14, although ISD-42 is lower depth than ISD-62, we found that ISD-42 has the best performance in validation. ISD using 42 depths is sufficient to deliver good performance and it is better in aspect of

resource effectiveness. We can observe that our compactness model with only 85K parameters achieves performance to 95.49% at mAP@0.50 in validation, which shows great potential for applications on computer vision system in real-time.

Table 4.7 Comparison with different depths and shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	**SC[2]**	**Param[3]**	**IoU[4]:0.50**	**IoU[4]:0.75**	**IoU[4]:0.50**	**IoU[4]:0.75**
24		15,038	84.03	44.10	86.84	49.72
	√	26,666	83.31	48.97	87.48	57.60
38		38,288	94.79	55.00	96.30	59.90
	√	69,776	95.24	57.14	95.42	68.67
42		46,700	88.19	50.64	91.35	59.25
	√	85,580	95.49	59.42	97.74	65.21
54		80,000	89.58	52.78	85.90	53.73
	√	148,832	93.40	50.12	94.17	58.55
62		106,472	83.99	42.55	84.80	47.82
	√	199,376	94.33	57.87	97.37	61.01

[1] Depth, [2] Shortcut, [3] Parameters (bytes), [4] Intersection over Union.

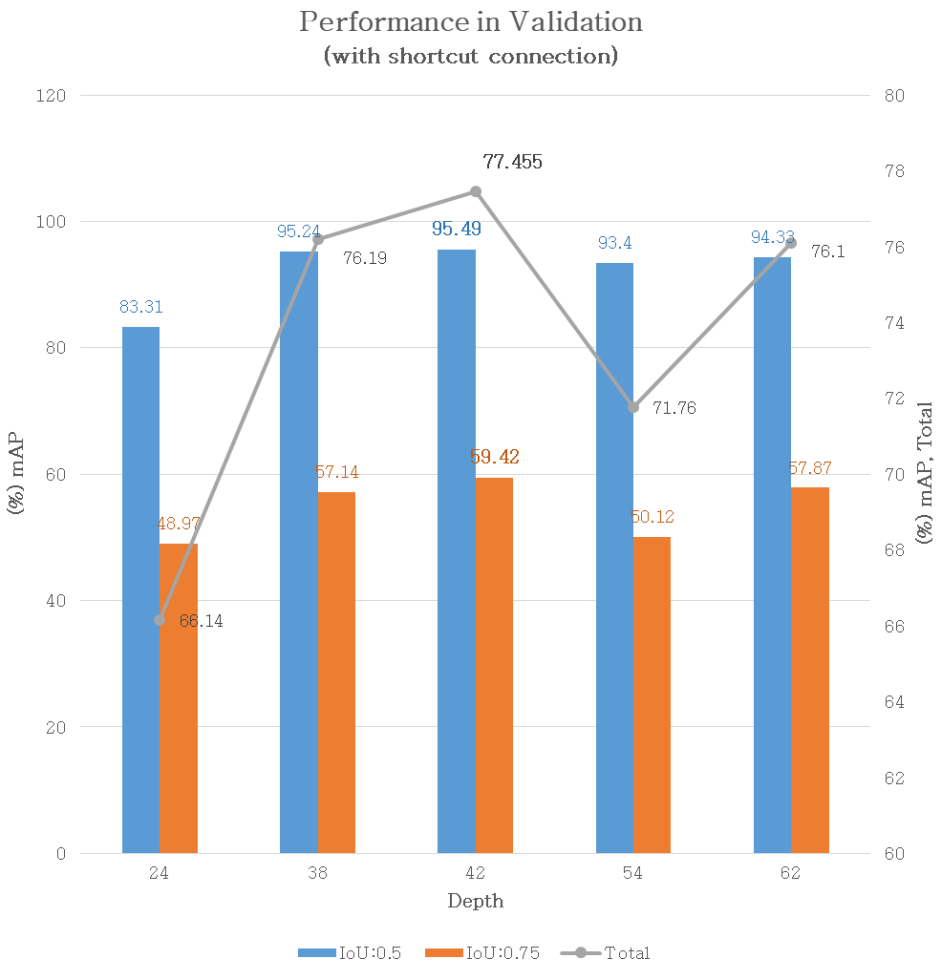


Figure 4.14 This chart shows performance in validation for comparison of different depths with shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

Shortcut connection. To address vanishing and exploding gradients, we propose our model with shortcut connection. Thus, we have experimented with and without shortcut connection for detecting suck-back state. The main results are summarized in Table 4.7. We observe that ISD with 62 depths using shortcut connection

significantly improves the performance from 42.55% to 57.87%, with a large margin (15.32%) at mAP@0.75 in validation.

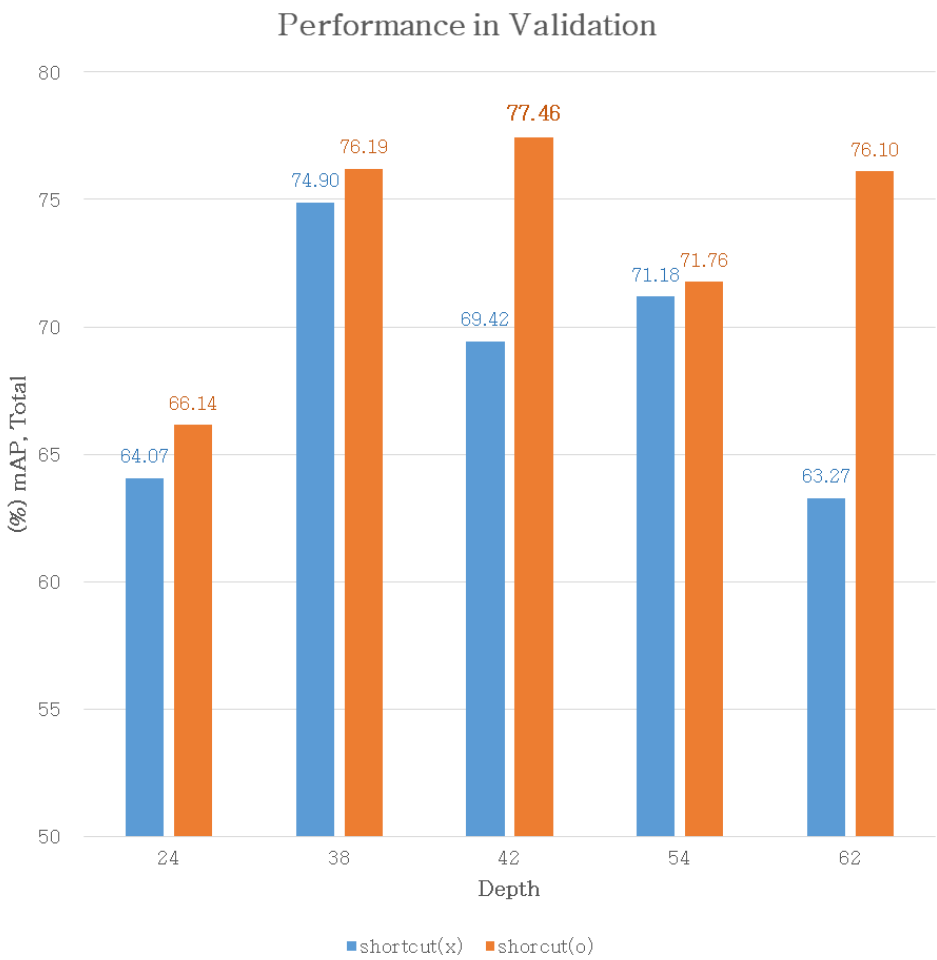


Figure 4.15 This chart shows performance in validation for comparison with different depths and shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

As illustrated in Figure 4.15, the experimental results show that ISD with shortcut connection has consistently better performance than ISD without shortcut connection. We empirically demonstrate that shortcut connection improves the performance by means of alleviating vanishing and exploding gradients, encouraging feature reuse.

Growth rate. We refer to the hyper-parameter k as the growth rate of the network. We show in Table 4.8 that a relatively small growth rate is sufficient to achieve better performance at mAP@0.50 in validation. However, as aforementioned, in order to reduce a huge number of training parameters increased by concatenation in dense block, ISD can apply three dynamic growth rate methods.

Table 4.8 Comparison of growth rate (k) with shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	G[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
38	6	69,776	95.24	57.14	95.42	68.67
	12	266,336	92.71	63.16	95.30	70.90
42	6	85,580	95.49	59.42	97.74	65.21
	12	281,672	95.14	53.10	95.52	61.52
54	6	148,832	93.40	50.12	94.17	58.55
	12	514,592	91.32	50.89	90.35	51.45

[1] Depth, [2] Growth rate (k), [3] Parameters (bytes), [4] Intersection over Union.

Table 4.9 Comparison of dynamic growth rate with shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 12.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	**G[2]**	**Param[3]**	**IoU[4]:0.50**	**IoU[4]:0.75**	**IoU[4]:0.50**	**IoU[4]:0.75**
	12A	266,336	92.71	63.16	95.30	70.90
38	12B	98,142	92.79	63.71	94.59	64.74
	12C	159,953	93.40	57.18	96.05	66.28
	12A	281,672	95.14	53.10	95.52	61.52
42	12B	104,553	91.20	44.12	87.97	45.03
	12C	179,968	95.66	55.85	96.81	65.30
	12A	514,592	91.32	50.89	90.35	51.45
54	12B	251,904	95.83	52.54	96.43	61.63
	12C	386,482	93.06	52.43	94.93	58.87

[1] Depth, [2] Growth rate (k), [3] Parameters (bytes), [4] Intersection over Union.

The first method increase or decrease the growth rate sequentially. In Table 4.9, we compare three options: (A) uniform growth rates (k,k,…,k) are used; (B) increasing growth rates (1,2,3,…,k) are used; (C) decreasing growth rates (k,k−1,k−2,…,2,1) are used. As illustrated in Table 4.9, we observe that ISD with 54 depths using increasing growth rates improves the performance from 91.32% to 95.83% at mAP@0.50 in validation, while requiring only 1/2 parameters.

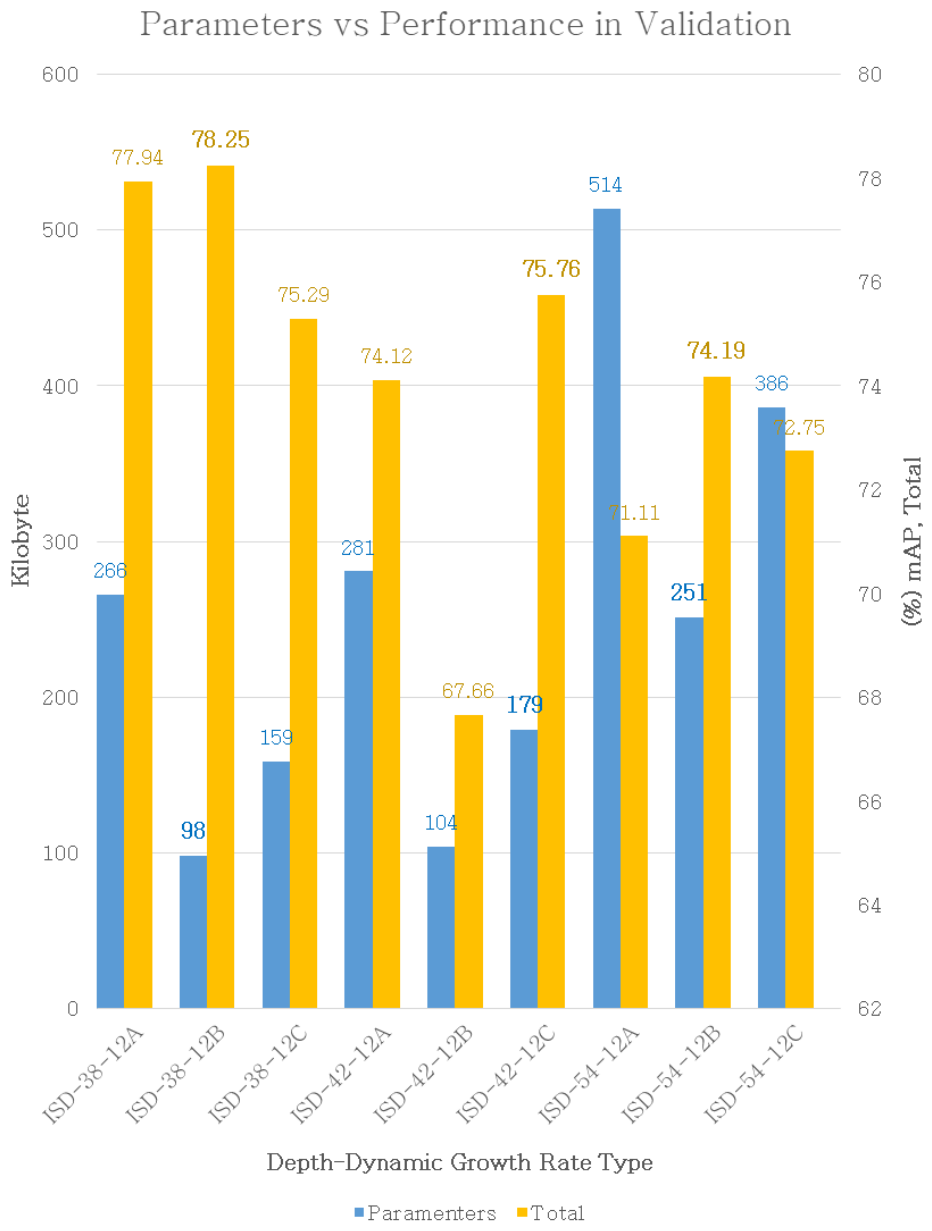


Figure 4.16 This chart shows performance in validation for comparison of various dynamic growth rate with shortcut connection in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 12.

As illustrated in Figure 4.16, we experimentally found that dynamic

growth rate improves the performance better than uniform growth rate, with more compactness. Note that it substantially reduces the number of training parameters.

The second is a method of compressing the generated feature map according to the growth rate. In this case, the growth rate is always fixed to one by the sum module or the mean module. The main results are summarized in Table 4.10. We experimentally found that training parameters can be remarkable reduced. However, the performance is not improved. At low depth, the sum method performs better than the mean method. Interestingly, at high depth, the mean method is better than the sum method.

Table 4.10 Comparison of sum and mean module in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	M[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
42	Mean	16,855	83.22	30.29	81.83	36.02
	Sum	16,855	88.16	34.79	87.06	40.85
74	Mean	34,140	85.76	32.78	84.10	38.87
	Sum	34,140	89.38	48.03	92.81	47.82
154	Mean	100,328	88.37	29.42	84.86	36.59
	Sum	100,328	73.60	20.39	73.37	29.26

[1] Depth, [2] Module, [3] Parameters (bytes), [4] Intersection over Union.

Table 4.11 Comparison of sum and mean module added to growth rate in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	M[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
42	Mean[+]	107,263	89.90	36.89	90.82	50.59
	Sum[+]	107,263	68.16	25.27	69.36	31.45
	None	85,580	95.49	59.42	97.74	65.21
	Mean	16,855	83.22	30.29	81.83	36.02
	Sum	16,855	88.16	34.79	87.06	40.85

[1] Depth, [2] Module, [3] Parameters (bytes), [4] Intersection over Union, [+]Add to growth rate

The third is a method designed to improve performance, by concatenating the compressed feature map obtained by sum or mean module and the feature map generated by growth rate. In this case, the growth rate is always one more due to the compressed feature map concatenation. The main results are summarized in Table 4.11. We found that the performance is not improved even with increasing training parameters.

Overfitting validation. In order to experimentally verify that our model is not overfitting, we evaluate performance with a new test dataset that has never been used for training, and compare results. We use new 194 images to verify overfitting in image segmentation of suck-back state.

Table 4.12 Comparison of results with different datasets in image segmentation of suck-back state. ISD has 42 depths with shortcut connection and the uniform growth rate (k) is 6.

Dataset	Test (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75
Existing dataset	95.49	59.42
New dataset	94.85	50.57

[1] Intersection over Union.

The main results are summarized in Table 4.12. Even if we evaluate performance with a new dataset that is completely different from the existing dataset, the performance at mAP@0.50 validation is almost the same. Therefore, we experimentally verified that our model is not overfitting in image segmentation of suck-back state.

4.3.2. Results on Dispensing State

In this section, in order to optimize our model with more compactness for detecting dispensing state, we justify the effectiveness of each design structure and parameters elaborated earlier. An example of image segmentation results on dispensing state is shown in Figure 4.17.

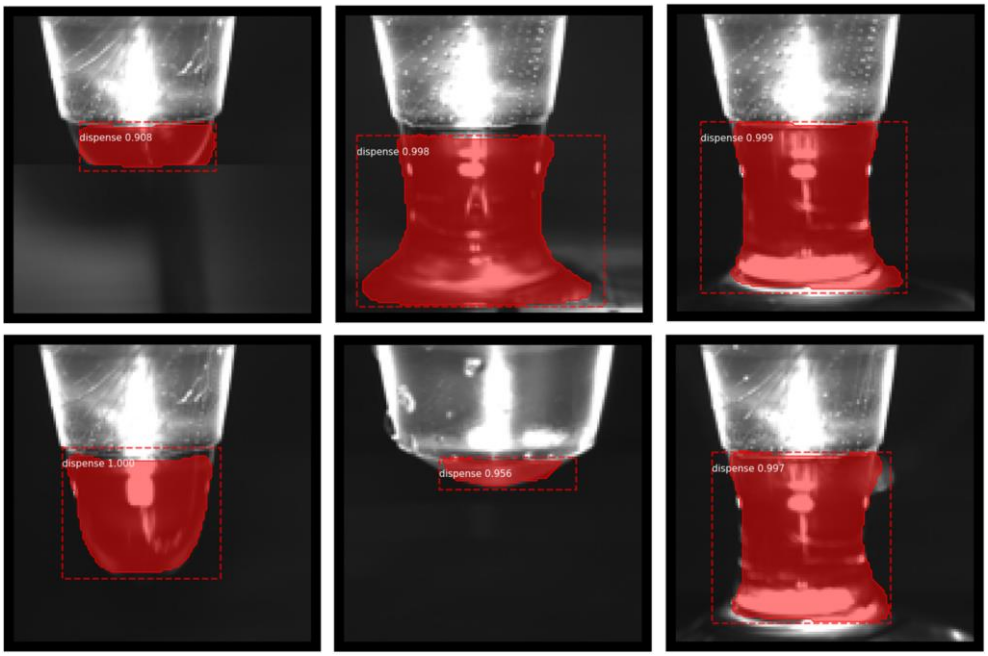


Figure 4.17 An example of image segmentation results on dispensing state.

The number of depth. We have experimented with various depths on ISD for detecting dispensing state. The main results are summarized in Table 4.13. We conjecture that the deeper layer is the better performance, as is well known. However, as shown in Figure 4.18, although ISD-38 is lower depth than ISD-62, we found that ISD-38 has the best performance in validation. ISD using 38 depths is sufficient to deliver good performance and it is better in aspect of resource effectiveness. We can observe that our compactness model with only 69K parameters achieves performance to 98.33% at mAP@0.50 in validation, which shows great potential for applications on computer vision system in real-time. Likewise results on suck-back state, we experimentally demonstrate that lower depth is

sufficient to good performance, even in case of dispensing state.

Table 4.13 Comparison with different depths and shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	SC[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
24		15,038	93.56	63.81	95.54	69.43
	√	26,666	94.00	69.10	95.18	77.14
30		23,048	94.22	71.10	92.78	65.57
	√	41,264	90.67	75.00	91.83	77.07
34		33,152	94.89	74.00	94.16	68.50
	√	60,560	95.67	73.67	96.87	77.50
38		38,288	95.33	66.67	93.45	70.61
	√	69,776	98.33	77.67	96.30	76.21
42		46,700	96.33	76.28	97.29	72.34
	√	85,580	97.56	76.33	97.01	80.68
44		51,146	92.33	76.67	93.59	75.38
	√	93,926	96.00	75.00	96.15	75.36
46		57,272	82.67	71.00	85.90	61.97
	√	105,632	95.83	64.22	95.16	62.96
50		72,560	92.00	64.33	92.12	62.82
	√	135,152	96.00	64.33	97.15	72.79
54		80,000	93.33	67.33	95.87	72.89
	√	148,832	96.00	70.00	96.15	69.18

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	SC[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
62		106,472	94.33	65.89	95.44	70.61
	√	199,376	94.67	77.00	97.67	75.38

[1] Depth, [2] Shortcut, [3] Parameters (bytes), [4] Intersection over Union.

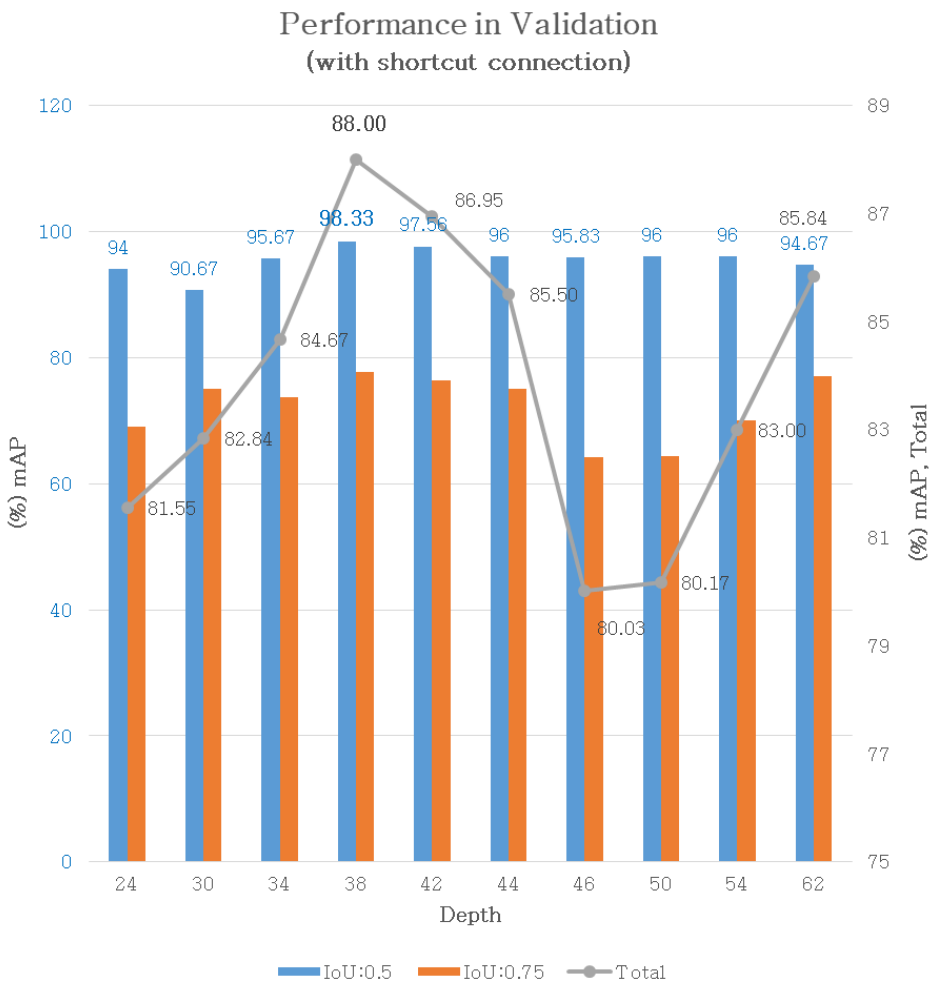


Figure 4.18 This chart shows performance in validation for comparison of different depths with shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

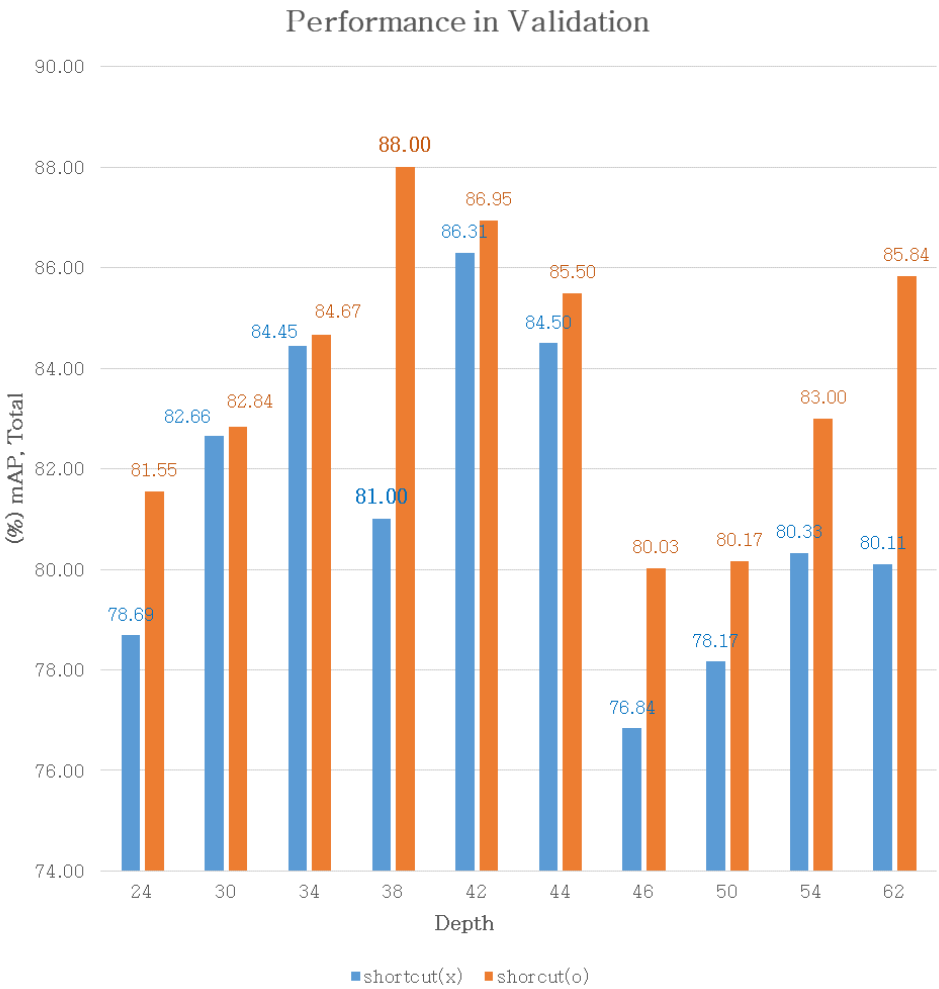


Figure 4.19 This chart shows performance in validation for comparison with different depths and shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 6.

Shortcut connection. To address vanishing and exploding gradients, we propose our model with shortcut connection. Thus, we have experimented with and without shortcut connection for detecting dispensing state. The main results are summarized in Table 4.13. We observe that ISD with 62 depths using shortcut connection

significantly improves the performance from 65.89% to 77.00%, with a large margin (11.11%) at mAP@0.75 in validation. Furthermore, we also observe that ISD with 42 depths using shortcut connection significantly improves the performance from 82.67% to 95.83%, with a large margin (13.16%) at mAP@0.50 in validation. As illustrated in Figure 4.19, the experimental results show that ISD with shortcut connection has consistently better performance than ISD without shortcut connection, as the same case of detecting suck-back state. As stated previously, it is especially notable that shortcut connection improves the performance by means of alleviating vanishing and exploding gradients, encouraging feature reuse.

Growth rate. We show in Table 4.14 that a relatively small growth rate is sufficient to achieve better performance in validation, even in results on dispensing state.

Table 4.14 Comparison of growth rate (k) with shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	G[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
38	6	69,776	98.33	77.67	96.30	76.21
	12	266,336	95.33	74.33	94.73	73.22
42	6	85,580	97.56	76.33	97.01	80.68
	12	281,672	94.00	70.56	93.88	63.32

[1] Depth, [2] Growth rate (k), [3] Parameters (bytes), [4] Intersection over Union.

As aforementioned, ISD use dynamic growth rate that applies different growth rates in each layer. In Table 4.15, we compare three options: (A) uniform growth rates (k,k,…,k) are used; (B) increasing growth rates (1,2,3,…,k) are used; (C) decreasing growth rates (k,k−1,k−2,…,2,1) are used; The main results are summarized in Table 4.15. We observe that ISD with 42 depths using decreasing growth rates improves the performance from 94.00% to 97.22% at mAP@0.50 in validation, while requiring only 3/5 parameters.

Table 4.15 Comparison of dynamic growth rate with shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 12.

ISD			Test (mAP, %)		Train (mAP, %)	
D[1]	G[2]	Param[3]	IoU[4]:0.50	IoU[4]:0.75	IoU[4]:0.50	IoU[4]:0.75
38	12A	266,336	95.33	74.33	94.73	73.22
	12B	98,142	97.33	74.00	96.82	69.73
	12C	159,953	97.47	71.52	96.72	72.57
42	12A	281,672	94.00	70.56	93.88	63.32
	12B	104,553	86.22	67.89	87.32	57.67
	12C	179,968	97.22	61.50	96.58	61.36
54	12A	514,592	96.33	72.67	96.11	72.37
	12B	251,904	95.67	71.28	93.88	69.47
	12C	386,482	97.67	72.33	98.29	80.91

[1] Depth, [2] Growth rate (k), [3] Parameters (bytes), [4] Intersection over Union.

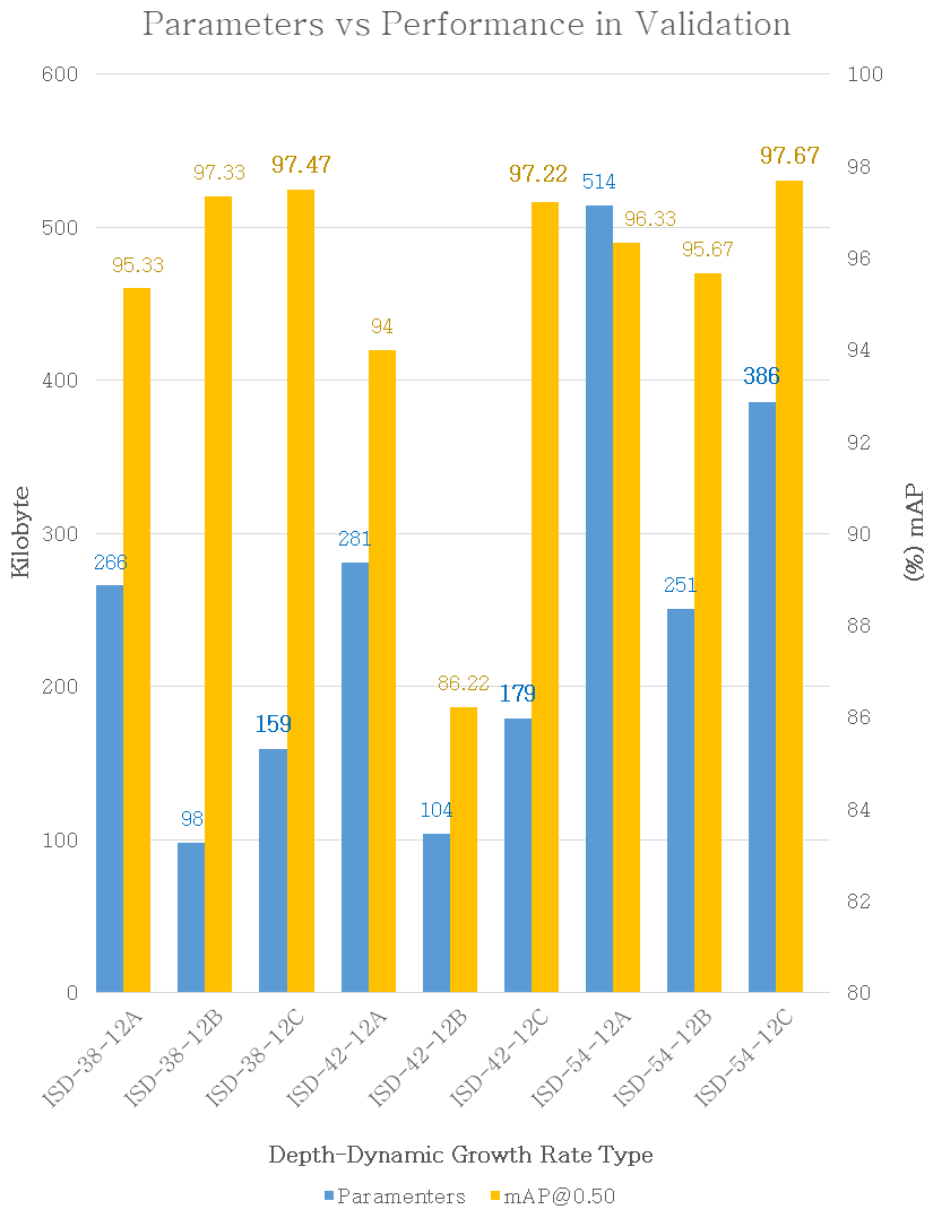


Figure 4.20 This chart shows performance in validation for comparison of various dynamic growth rate with shortcut connection in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The uniform growth rate (k) is 12.

As illustrated in Figure 4.20, likewise results on suck-back state

even in dispensing state, we experimentally demonstrate that dynamic growth rate consistently improves the performance better than uniform growth rate, with more compactness. Note that it substantially reduces the number of training parameters.

Overfitting validation. In order to experimentally verify that our model is not overfitting, we evaluate performance with a new test dataset that has never been used for training and compare results. We use new 210 images to verify overfitting in image segmentation of dispensing state. The main results are summarized in Table 4.16. Even if we evaluate performance with a new dataset that is completely different from the existing dataset, the performance at mAP@0.50 validation is almost the same. Therefore, we experimentally verified that our model is not overfitting in image segmentation of dispensing state.

Table 4.16 Comparison of results with different datasets in image segmentation of dispensing state. ISD has 38 depths with shortcut connection and the uniform growth rate (k) is 6

Dataset	Test (mAP, %)	
	IoU[1]:0.50	IoU[1]:0.75
Existing dataset	98.33	77.67
New dataset	98.10	91.59

[1] Intersection over Union.

4.4. Comparison with State-of-the-art Methods

In this section, we compare our model with state-of-the-art backbone networks of Mask R-CNN framework. Thus, we empirically demonstrate the compactness and performance of our model. Our experiments are conducted on object categories which are suck-back state and dispensing state respectively.

Results on suck-back state. The main results are summarized in Table 4.17. ISD achieves consistently better results than stat-of-the-art methods with much more compactness structure. Especially, our ISD-38 achieves 95.24% at mAP@0.50 in validation, which outperforms the baseline DenseNet-38 with a large margin (16.97%), while requiring only 1/4 parameters. We also observe that ISD-38 can achieve comparable better results at mAP@0.75 than ResNet-38 requiring a huge memory space to store the massive parameters, with only much smaller 1/268 parameters, which shows great potential for application on resource bounded devices.

As the size of the network increases, the inference and the training become slower and require more data. There is generally a trade-off between performance and speed. When one needs real-time detectors, like for computer vision, one loses some precision. In Table 4.17, the highest result of 96.59% at mAP@0.50 in validation

are obtained with ResNet-38. Our ISD-42 achieves 95.49% at mAP@0.50 in validation, 1.1% lower. However, as shown in Table 4.18, the number of parameters is improved significantly by 217 times. The running time is also improved by 7 times. Furthermore interestingly, our ISD-42 is 3.45% higher than ResNet-38 at mAP@0.75 in validation. As illustrated in Figure 4.21, the experimental results show that our model achieves compactness and performance balance for embedded real-time application.

Table 4.17 Comparison with state-of-the-art backbone networks in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6.

BN[1]	Param[2]	Test (mAP, %)		Train (mAP, %)	
		IoU[3]:0.50	IoU[3]:0.75	IoU[3]:0.50	IoU[3]:0.75
ResNet-26	14,008K	94.82	61.37	98.76	75.47
ResNet-38	18,496K	96.59	55.97	96.93	78.57
ResNet-50	23,604K	93.36	50.58	95.49	67.45
ResNet-101	42,674K	94.85	51.58	97.60	65.14
DenseNet-24	98K	78.77	36.04	84.33	48.81
DenseNet-38	253K	78.27	35.20	82.01	42.98
DenseNet-42	308K	85.76	40.74	86.28	42.70
DenseNet-54	516K	84.72	35.71	82.90	40.53
DenseNet-62	681K	85.59	37.57	85.43	46.35
ISD-24	26K	83.31	48.97	87.48	57.60
ISD-38	69K	95.24	57.14	95.42	68.67

BN[1]	Param[2]	Test (mAP, %)		Train (mAP, %)	
		IoU[3]:0.50	IoU[3]:0.75	IoU[3]:0.50	IoU[3]:0.75
ISD-42	85K	95.49	59.42	97.74	65.21
ISD-54	148K	93.40	50.12	94.17	58.55
ISD-62	199K	94.33	57.87	97.37	61.01

[1] Backbone Network, [2] Parameters (kilobyte), [3] Intersection over Union.

Table 4.18 Comparison of state-of-the-art backbone networks with FLOPs and running time in suck-back state. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6. Input size of backbone network is 120×120. The running time is the processing time of CPU per image.

BN[1]	Param[2]	MFLOPs			Time[4]	Performance
		Mul[3]	Add	Total		mAP@0.50/0.75
ResNet-26	14,008K	83.87	55.87	209.81	110.68	94.82/61.37
ResNet-38	18,496K	110.72	73.75	276.99	148.67	96.59/55.97
ResNet-50	23,604K	141.31	94.12	353.50	193.09	93.36/50.58
ResNet-101	42,674K	255.42	170.12	639.01	358.52	94.85/51.58
DenseNet-24	98K	0.96	0.19	1.65	23.68	78.77/36.04
DenseNet-38	253K	2.49	0.50	4.26	37.63	78.27/35.20
DenseNet-42	308K	3.04	0.60	5.19	42.69	85.76/40.74
DenseNet-54	516K	5.10	1.02	8.70	55.13	84.72/35.71
DenseNet-62	681K	6.74	1.34	11.48	65.31	85.59/37.57
ISD-24	26K	0.23	0.05	0.40	13.50	83.31/48.97
ISD-38	69K	0.67	0.13	1.15	18.38	95.24/57.14
ISD-42	85K	0.77	0.16	1.33	21.38	95.49/59.42
ISD-54	148K	1.36	0.29	2.34	27.77	93.40/50.12
ISD-62	199K	1.83	0.39	3.15	31.09	94.33/57.87

[1] Backbone Network, [2] Parameters (kilobyte), [3] Multiply, [4] Running Time (millisecond).

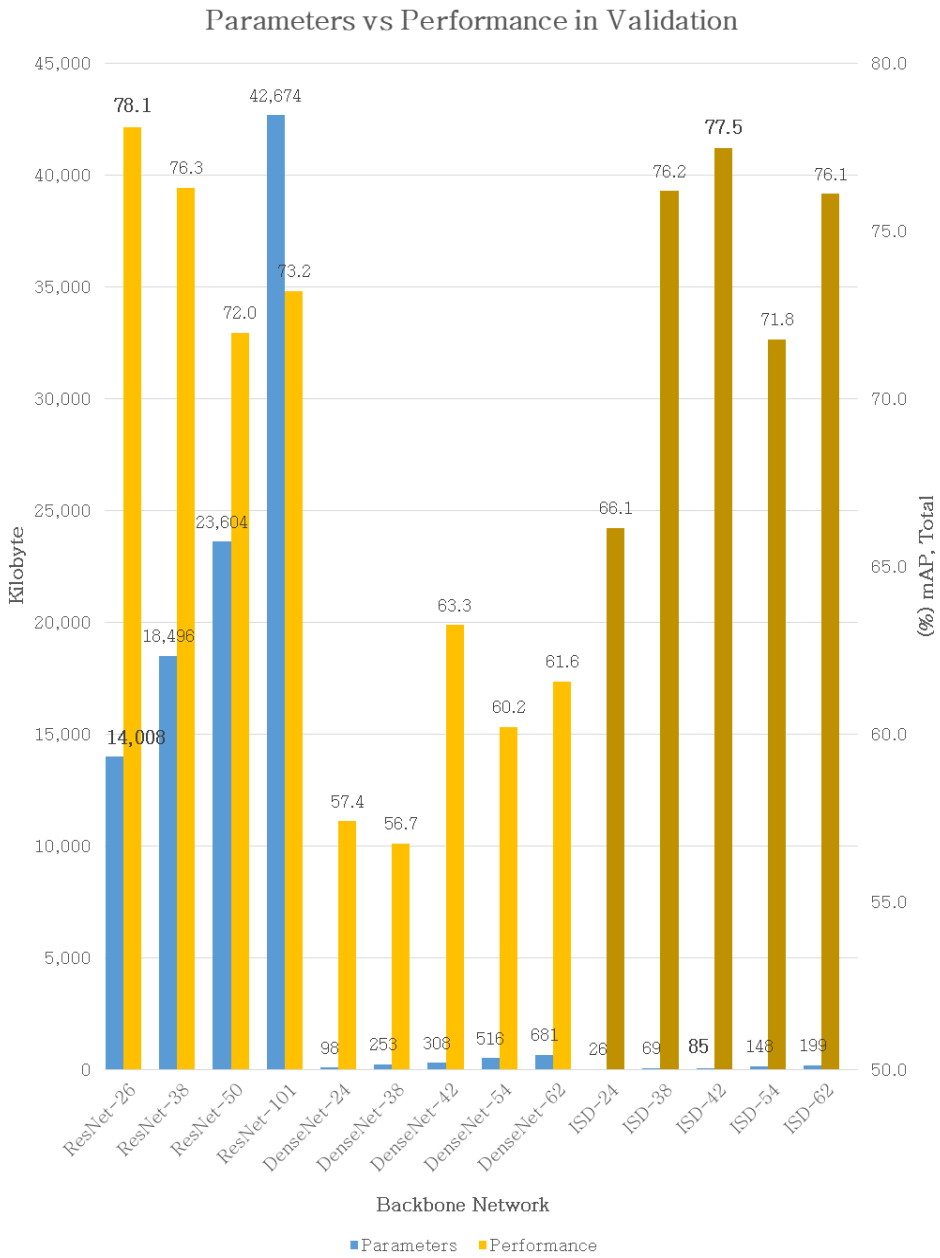


Figure 4.21 This shows performance in validation for comparison with various state-of-the-art backbone networks in suck-back state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6.

Results on dispensing state. The main results are summarized in Table 4.19. Likewise results on suck-back state, on aspect of performance versus memory, ISD achieves consistently better results than stat-of-the-art methods with much more compactness structure, even in dispensing state.

Table 4.19 Comparison with state-of-the-art backbone networks in dispensing state results. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6.

BN[1]	Param[2]	Test (mAP, %)		Train (mAP, %)	
		IoU[3]:0.50	IoU[3]:0.75	IoU[3]:0.50	IoU[3]:0.75
ResNet-26	14,008K	98.67	85.06	99.15	84.76
ResNet-101	42,674K	99.67	77.13	99.86	79.87
DenseNet-24	98K	84.00	58.00	76.88	50.00
DenseNet-38	253K	81.39	62.67	85.04	58.55
DenseNet-42	308K	88.00	60.00	83.71	56.13
DenseNet-54	516K	78.67	58.00	78.35	55.27
DenseNet-62	681K	85.33	63.67	86.18	62.77
ISD-24	26K	94.00	69.10	95.18	77.14
ISD-38	69K	98.33	77.67	96.30	76.21
ISD-42	85K	94.00	69.10	95.18	77.14
ISD-54	148K	96.00	70.00	96.15	69.18
ISD-62	199K	94.67	77.00	97.67	75.38

[1] Backbone Network, [2] Parameters (kilobyte), [3] Intersection over Union.

Table 4.20 Comparison of state-of-the-art backbone networks with FLOPs and running time in dispensing state. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6. Input size of backbone network is 120×120. The running time is the processing time of CPU per image.

BN[1]	Param[2]	MFLOPs			Time[4]	Performance
		Mul[3]	Add	Total		mAP@0.50/0.75
ResNet-26	14,008K	83.87	55.87	209.81	110.68	98.67/85.06
ResNet-101	42,674K	255.42	170.12	639.01	358.52	99.67/77.13
DenseNet-24	98K	0.96	0.19	1.65	23.68	84.00/58.00
DenseNet-38	253K	2.49	0.50	4.26	37.63	81.39/62.67
DenseNet-42	308K	3.04	0.60	5.19	42.69	88.00/60.00
DenseNet-54	516K	5.10	1.02	8.70	55.13	78.67/58.00
DenseNet-62	681K	6.74	1.34	11.48	65.31	85.33/63.67
ISD-24	26K	0.23	0.05	0.40	13.50	94.00/69.10
ISD-38	69K	0.67	0.13	1.15	18.38	98.33/77.67
ISD-42	85K	0.77	0.16	1.33	21.38	94.00/69.10
ISD-54	148K	1.36	0.29	2.34	27.77	96.00/70.00
ISD-62	199K	1.83	0.39	3.15	31.09	94.67/77.00

[1] Backbone Network, [2] Parameters (kilobyte), [3] Multiply, [4] Running Time (millisecond).

Particularly, our ISD-38 achieves 98.33% at mAP@0.50 in validation, which outperforms the baseline DenseNet-38 with a large margin (16.94%), while requiring only 1/4 parameters. We also observe that ISD-38 can achieve comparable better results at mAP@0.75 than ResNet-101 requiring a huge memory space to store the massive parameters, with only much smaller 1/625 parameters, which shows great potential for application on resource bounded devices. In Table

4.19, the highest result of 99.67% at mAP@0.50 in validation are obtained with ResNet-101. In the meanwhile, our ISD-38 achieves 98.33% at mAP@0.50 in validation, 1.34% lower. However, as shown in Table 4.20, the number of parameters is improved significantly by 625 times. The running time is also improved by 20 times. Furthermore interestingly, our ISD-38 is 0.54% higher than ResNet-101 at mAP@0.75 in validation. As illustrated in Figure 4.22, ResNet-26 is 7.73% higher than ISD-38 at total performance in validation. However, ISD-38 is approximately 200 times more compact than ResNet-26 at parameters. The speed is also 6 times faster. Additionally, there is a little difference (0.34%) between ISD-38 and ResNet-26 in performance mAP@0.50 validation. Thus, the experimental results show that our model consistently achieves compactness and performance balance for embedded real-time application, as the same results on suck-back state. Model compactness in terms of the number of parameters, and performance is an important trade-off for various applications of deep neural networks in detection. ResNet which is the most common backbone network of Mask R-CNN framework, require a huge memory space to store the massive parameters.

Therefore the models are usually unsuitable for low-end devices like embedded computer vision system in industry. Thanks to the parameter-efficient dense block with shortcut connection, our model significantly is much smaller with high performance than most competitive methods.

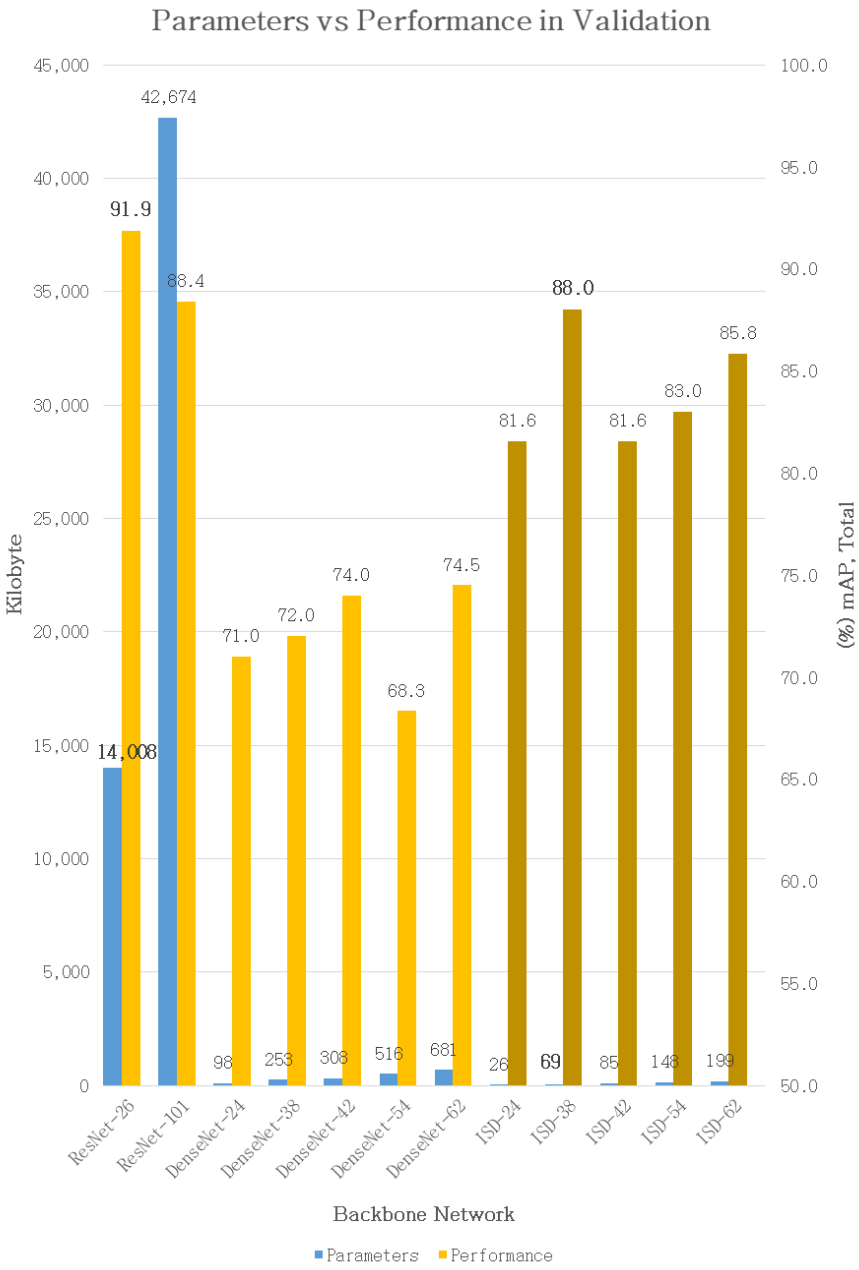


Figure 4.22 This shows performance in validation for comparison with various state-of-the-art backbone networks in dispensing state results. Additionally, the training process failed to converge for ResNet-50. We experiment with model weights having the lowest validation loss obtained during the training up to 100 epochs. The growth rate of DenseNet is 12. The uniform growth rate of ISD is 6.

Chapter 5. Conclusion

This dissertation aimed to improve defect detection in semiconductor photolithography inspection systems. We investigated three questions. First, is it possible to train image segmentation networks from scratch directly with only smaller dataset without the pre-trained models? Second, are there any principles to design a resource efficient network structure for image segmentation, meanwhile keeping high detection accuracy? Third, is there any methodology to improve inspection performance other than network design? To achieve our main goal, we proposed a novel deep architecture that can be used as a backbone network of the image segmentation, and applied image filtering method as pre-processing to training for improving performance.

In general, since most of computer vision operated in semiconductor photolithography inspection systems use image processing algorithm, inspection faults easily occur even with small

changes in the external environment. On the other hand, deep learning has a strong characteristic of classifying images even with these external changes. However, since the features extracted from image for inspection in semiconductor photolithography are limited to a specific area of image, there is a limit to extracting them with only the feature map of the CNN. No matter how much model is trained with a lot of various images for image segmentation, fault still occurs in detecting the mask region. Therefore, we combined the advantages of image processing algorithm and the advantages of deep learning, in order to achieve remarkable performance.

We performed this study with step by step. We applied the existing Mask R-CNN framework to photolithography inspection systems without modification of backbone network. Next, we changed ResNet, which is used as backbone network in Mask R-CNN framework, to DenseNet. The reason is that DenseNet is more efficient and compressed model than ResNet. Then, we designed a novel deep architecture based on DenseNet, designed image filter for pre-processing to improve performance, and experimentally demonstrated the performance of our model. In detail, we performed this study as follows.

Firstly, we have analyzed the inspection method of computer vision used in currently operating semiconductor photolithography process for defect detection. Then, we have studied various the state-of-the-art deep learning based instance segmentation, backbone structure, and the improved network structure for

extracting enhanced feature map.

Secondly, we designed a novel deep architecture called as ISD (Image Segmentation Detector) to meet our goal based on related studies. Our architecture has the same concept of combining feature re-usage ability of ResNet and feature re-exploration ability of DenseNet as in DPN and MixNet, However, the structure is significantly different from other networks. Notably, in order to achieve superior efficiency with compactness, we combined feature re-usage with same size in each block which is a group of layers and feature re-exploration with dynamic growth rate in each layer. In other words, ISD improves DenseNet by applying a dynamic growth rate to reduce the number of parameters, and by using shortcut connection in each block to alleviate the gradient vanishing problem and to achieve superior efficiency with compactness. Furthermore, ISD performs down-sampling in a dense block rather than a transition layer, and changes pre-activation to post-activation. Additionally, ISD removes pooling of transition layer to transfer more information, and reduces the bottleneck width for more compactness.

Thirdly, in order to improve performance of image segmentation in semiconductor photolithography inspection systems, we designed image filter which is composed of a pair of 3×3 convolution masks for extracting features by emphasizing boundary. From this study, we concluded that our image filter, which is used for pre-processing in training, play an important role in extracting the enhanced feature-map for image segmentation.

Lastly, model optimization and performance are an important trade-off for the applications of deep neural networks in image segmentation tasks for real-time application. Thus, in order to optimize ISD, we conducted various experiments with three cases which are the number of depth, shortcut connection, and dynamic growth rate. Our experiments was conducted on object categories which are suck-back state and dispensing state respectively. Then, we compared our model with state-of-the-art backbone networks of Mask R-CNN framework. In image segmentation of suck-back state, we experimentally demonstrated that our ISD-42 significantly outperforms state-of-the-art DenseNet-42 in terms of both accuracy (9.73% more accurate) and parameters (3 times less) at mAP@0.50 in validation. Furthermore, our ISD-42 improves 217 times smaller in the number of parameters, and 3.45% higher accurate than state-of-the-art ResNet-38 at mAP@0.75 in validation. Furthermore, the running speed is also improved by 7 times. In image segmentation of dispensing state, we experimentally demonstrated that our ISD-38 achieves 98.33% at mAP@0.50 in validation, which outperforms the baseline DenseNet-38 with a large margin (16.94%), while requiring only 1/4 parameters. We also observed that ISD-38 can achieve comparable better results at mAP@0.75 than ResNet-101 requiring a huge memory space to store the massive parameters, with only much smaller 1/625 parameters, which shows great potential for application on resource bounded devices. Furthermore, the running speed is also improved

by 20 times. Therefore, we experimentally demonstrated that ISD can be applicable to many image segmentation architecture to achieves the right speed (parameters) and accuracy balance for a given application and platform.

This dissertation presents a novel deep architecture (backbone network), the ISD, to tackle the problem that training dataset limited in specific industry domain such as semiconductor photolithography might cause overfitting at training and quality mismatch at inference. Particularly, our model which acts as the main feature extractor, is more compact with higher performance than most competitive models. Furthermore, compactness of our model is suitable for application on resource bounded devices due to addressing real-time problem. Our model is simple to construct and can be trained directly on full images. Our proposed approach, which is learning deep models from scratch, has very appealing advantages over existing solutions. Especially, our approach is suitable for image segmentation of industry domain which does not have large-scale image dataset like ImageNet for transfer learning. According to our method including pre-processing, enhanced feature-maps can be obtained for image segmentation.

In conclusion, the most important contribution of this work is probably the first application of deep learning (image segmentation) technology to semiconductor photolithography inspection systems. Furthermore, we proposed an efficient novel deep architecture (backbone network) that can be used learning from scratch in specific domains where the acquisition of training images is limited, such as

semiconductor photolithography. Thus, we believe that it can be useful to many future image segmentation research efforts in diverse industry domain which is requiring real-time and good performance with only smaller training dataset.

Bibliography

[1] A. M. Hafiz and G. M. Bhat, "A survey on instance segmentation: state of the art," *International journal of multimedia information retrieval,* vol. 9, no. 3, pp. 171-189, 2020, doi: 10.1007/s13735-020-00195-x.

[2] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied soft computing,* vol. 70, pp. 41-65, 2018, doi: 10.1016/j.asoc.2018.05.018.

[3] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic Segmentation," arXiv:1801.00868v3, 2018.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, 2015, doi: 10.1038/nature14539.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv.org,* 2014.

[6] R. Girshick, "Fast R-CNN," *Microsoft Research,* 2015.

[7] R. Shaoqing, H. Kaiming, R. Girshick, and S. Jian, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 6, pp. 1137-1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[8] K. He, G. Gkioxari, P. Doll´ar, and R. Girshick, "Mask R-CNN," *Facebook AI Research (FAIR),* 2018.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," ed: IEEE, 2016, pp. 779-788.

[10] L. Jiao *et al.*, "A Survey of Deep Learning-Based Object Detection," *IEEE access,* vol. 7, pp. 128837-128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[11] M. Tiwari, S. S. Lamba, and B. Gupta, "An image processing and computer vision framework for efficient robotic sketching," *Procedia Computer Science,* vol. 133, pp. 284-289, 2018, doi: 10.1016/j.procs.2018.07.035.

[12] J. Byung-Wan and L. Yun-Sung, "Computer Vision-Based Bridge Displacement Measurements Using Rotation-Invariant Image Processing Technique," *Sustainability,* vol. 10, no. 6, p. 1785, 2018, doi: 10.3390/su10061785.

[13] P. K. Saha, "Tensor scale: A local morphometric parameter with applications to computer vision and image processing," *Computer*

Vision and Image Understanding, vol. 99, no. 3, pp. 384-413, 2005, doi: 10.1016/j.cviu.2005.03.003.

[14] N. Gonçalves, V. Carvalho, M. Belsley, R. M. Vasconcelos, F. O. Soares, and J. Machado, "Yarn features extraction using image processing and computer vision – A study with cotton and polyester yarns," *Measurement,* vol. 68, pp. 1-15, 2015, doi: 10.1016/j.measurement.2015.02.010.

[15] C. Ng, C. Wu, W. Ip, C. Chan, and T. Ho, "A real time quality monitoring system for the lighting industry : a practical and rapid approach using computer vision and image processing (CVIP) tools," *International journal of engineering business management,* vol. 3, no. 4, pp. 14-21, 2011, doi: 10.5772/45670.

[16] D. L. B. R. Jurjo, C. Magluta, N. Roitman, and P. Batista Gonçalves, "Analysis of the structural behavior of a membrane using digital image processing," *Mechanical Systems and Signal Processing,* vol. 54-55, pp. 394-404, 2015, doi: 10.1016/j.ymssp.2014.08.010.

[17] S. Sunoj *et al.*, "Sunflower floral dimension measurements using digital image processing," *Comput. Electron. Agric.,* vol. 151, pp. 403-415, 2018, doi: 10.1016/j.compag.2018.06.026.

[18] H.-S. Choi, J.-H. Cheung, S.-H. Kim, and J.-H. Ahn, "Structural dynamic displacement vision system using digital image processing," *NDT and E International,* vol. 44, no. 7, pp. 597-608, 2011, doi: 10.1016/j.ndteint.2011.06.003.

[19] A. Iswardani and W. Hidayat, "Mammographic Image Enhancement

using Digital Image Processing Technique," *arXiv.org,* 2018.

[20] S. Nur Mutiara, R. Pola, and D. Tresna, "Vision-Based Pipe Monitoring Robot For Crack Detection Using Canny Edge Detection Method as an Image Processing Technique," *Kinetik,* vol. 2, no. 4, pp. 243-250, 2017, doi: 10.22219/kinetik.v2i4.243.

[21] L. Liu *et al.*, "Deep Learning for Generic Object Detection: A Survey," 2018.

[22] S. Agarwal and F. Jurie, "Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks," *arXiv.org,* 2019.

[23] D. Jia, D. Wei, R. Socher, L. Li-Jia, L. Kai, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," ed, 2009, pp. 248-255.

[24] P. Sinno Jialin and Y. Qiang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 22, no. 10, pp. 1345-1359, 2010, doi: 10.1109/TKDE.2009.191.

[25] S. Xia *et al.*, "Transferring Ensemble Representations Using Deep Convolutional Neural Networks for Small-Scale Image Classification," *IEEE Access,* vol. 7, no. 99, pp. 168175-168186, 2019, doi: 10.1109/ACCESS.2019.2912908.

[26] W. Cui, G. Zheng, Z. Shen, S. Jiang, and W. Wang, "Transfer Learning for Sequences via Learning to Collocate," *arXiv.org,* 2019.

[27] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan, "A theory of learning from different domains," *Machine Learning,* vol. 79, no. 1-2, pp. 151-175, 2010, doi: 10.1007/s10994-

009-5152-4.

[28] Y. Ganin *et al.*, "Domain-Adversarial Training of Neural Networks," *arXiv.org,* 2016.

[29] P. Sinno Jialin, I. W. Tsang, J. T. Kwok, and Y. Qiang, "Domain Adaptation via Transfer Component Analysis," *IEEE Transactions on Neural Networks,* vol. 22, no. 2, pp. 199-210, 2011, doi: 10.1109/TNN.2010.2091281.

[30] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *arXiv.org,* 2016.

[31] W. Liu, D. Anguelov, C. Szegedy, S. Reed, F. Cheng-Yang, and A. Berg, "SSD: Single Shot MultiBox Detector," vol. 9905, ed. Ithaca, 2016.

[32] L. Tsung-Yi, P. Goyal, R. Girshick, H. Kaiming, and P. Dollar, "Focal Loss for Dense Object Detection," vol. 2017-, ed, 2017, pp. 2999-3007.

[33] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection," *arXiv.org,* 2016.

[34] S. Bell, C. Zitnick, K. Bala, and R. Girshick, "Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks," *arXiv.org,* 2015.

[35] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu, and Y. Chen, "RON: Reverse Connection with Objectness Prior Networks for Object Detection," *arXiv.org,* 2017.

[36] P. Chao *et al.*, "MegDet: A Large Mini-Batch Object Detector," *arXiv.org,* 2018.

[37] B. Singh and L. Davis, "An Analysis of Scale Invariance in Object Detection - SNIP," *arXiv.org,* 2018.

[38] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation Networks for Object Detection," *arXiv.org,* 2018.

[39] H. Xu, X. Lv, X. Wang, R. Zhou, N. Bodla, and R. Chellappa, "Deep Regionlets for Object Detection," *arXiv.org,* 2018.

[40] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A Real-Time Object Detection System on Mobile Devices," arXiv:1804.06882v3, 2018.

[41] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 4, pp. 640-651, 2017, doi: 10.1109/TPAMI.2016.2572683.

[42] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for Object Segmentation and Fine-grained Localization," *arXiv.org,* 2015.

[43] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," *arXiv.org,* 2016.

[44] Y. Fisher and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," *arXiv.org,* 2016.

[45] J. Wang and A. K. Asundi, "A computer vision system for wineglass

defect inspection via Gabor-filter-based texture features," *Information Sciences,* vol. 127, no. 3, pp. 157-171, 2000, doi: 10.1016/S0020-0255(00)00036-0.

[46] J. Wang, Y. Liu, D. Zhang, H. Peng, and Y. Zhu, "A new computer vision based multi-indentation inspection system for ceramics," *An International Journal,* vol. 76, no. 2, pp. 2495-2513, 2017, doi: 10.1007/s11042-015-3223-z.

[47] M. Quintana, J. Torres, and J. M. Menendez, "A Simplified Computer Vision System for Road Surface Inspection and Maintenance," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 3, pp. 608-619, 2016, doi: 10.1109/TITS.2015.2482222.

[48] A. l. Gonzalez *et al.*, "Automatic Traffic Signs and Panels Inspection System Using Computer Vision," *IEEE Transactions on Intelligent Transportation Systems,* vol. 12, no. 2, pp. 485-499, 2011, doi: 10.1109/TITS.2010.2098029.

[49] R. G. Saeidi, M. Latifi, S. S. Najar, and A. G. Saeidi, "Computer Vision-Aided Fabric Inspection System for On-Circular Knitting Machine," *Textile Research Journal,* vol. 75, no. 6, pp. 492-497, 2005, doi: 10.1177/0040517505053874.

[50] J. Lopez, M. Cobos, and E. Aguilera, "Computer-based detection and classification of flaws in citrus fruits," *Neural Computing and Applications,* vol. 20, no. 7, pp. 975-981, 2011, doi: 10.1007/s00521-010-0396-2.

[51] R. Seulin, F. Merienne, and P. Gorria, "Simulation of Specular Surface

Imaging Based on Computer Graphics: Application on a Vision Inspection System," *EURASIP Journal on Advances in Signal Processing,* vol. 2002, no. 7, pp. 1-10, 2002, doi: 10.1155/S1110865702203030.

[52] N. C. Mithun, N. U. Rashid, and S. M. M. Rahman, "Detection and Classification of Vehicles From Video Using Multiple Time-Spatial Images," *IEEE transactions on intelligent transportation systems,* vol. 13, no. 3, pp. 1215-1225, 2012, doi: 10.1109/TITS.2012.2186128.

[53] J. Wang, H. Zheng, Y. Huang, and X. Ding, "Vehicle Type Recognition in Surveillance Images From Labeled Web-Nature Data Using Deep Transfer Learning," *IEEE transactions on intelligent transportation systems,* vol. 19, no. 9, pp. 2913-2922, 2018, doi: 10.1109/TITS.2017.2765676.

[54] J. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective Search for Object Recognition," *International journal of computer vision,* vol. 104, no. 2, pp. 154-171, 2013, doi: 10.1007/s11263-013-0620-5.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," arXiv:1406.4729v4, 2014.

[56] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," arXiv:1612.03144v2 , 2016.

[57] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully Convolutional Instance-

aware Semantic Segmentation," arXiv:1611.07709v2, 2016.

[58] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to Segment Object Candidates," arXiv:1506.06204v2, 2015.

[59] J. Dai, K. He, and J. Sun, "Instance-aware Semantic Segmentation via Multi-task Network Cascades," arXiv:1512.04412v1, 2015.

[60] A. Arnab and P. H. S. Torr, "Bottom-up Instance Segmentation using Deep Higher-Order CRFs," arXiv:1609.02583v1, 2016.

[61] A. Fathi *et al.*, "Semantic Instance Segmentation via Deep Metric Learning," arXiv:1703.10277v1, 2017.

[62] X. Liang, L. Lin, Y. Wei, X. Shen, J. Yang, and S. Yan, "Proposal-Free Network for Instance-Level Object Segmentation," *IEEE Trans Pattern Anal Mach Intell,* vol. 40, no. 12, pp. 2978-2991, 2018, doi: 10.1109/TPAMI.2017.2775623.

[63] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother, "InstanceCut: from Edges to Instances with MultiCut," arXiv:1611.08272v1, 2016.

[64] M. Bai and R. Urtasun, "Deep Watershed Transform for Instance Segmentation," arXiv:1611.08303v2, 2016.

[65] L. Shu, J. Jiaya, S. Fidler, and R. Urtasun, "SGN: Sequential Grouping Networks for Instance Segmentation," ed: IEEE, 2017, pp. 3516-3524.

[66] A. Arnab and P. H. S. Torr, "Pixelwise Instance Segmentation with a Dynamically Instantiated Network," arXiv:1704.02386v1, 2017.

[67] Y. Liu *et al.*, "Affinity Derivation and Graph Merge for Instance Segmentation," arXiv:1811.10870v1, 2018.

[68] N. Gao, Y. Shan, Y. Wang, X. Zhao, and K. Huang, "SSAP: Single-Shot Instance Segmentation With Affinity Pyramid," *IEEE transactions on circuits and systems for video technology,* vol. 31, no. 2, pp. 661-673, 2021, doi: 10.1109/TCSVT.2020.2985420.

[69] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," arXiv:1604.01685v2, 2016.

[70] D. Bolya, C. Zhou, F. Xiao, and Y. Lee, "YOLACT: Real-time Instance Segmentation," *arXiv.org,* 2019.

[71] W. Xu, H. Wang, F. Qi, and C. Lu, "Explicit Shape Encoding for Real-Time Instance Segmentation," arXiv:1908.04067v1, 2019.

[72] E. Xie *et al.*, "PolarMask: Single Shot Instance Segmentation with Polar Representation," arXiv:1909.13226v4, 2019.

[73] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully Convolutional One-Stage Object Detection," arXiv:1904.01355v5, 2019.

[74] J. Ahn, S. Cho, and S. Kwak, "Weakly Supervised Learning of Instance Segmentation with Inter-pixel Relations," arXiv:1904.05044v3, 2019.

[75] D. Neven, B. De Brabandere, M. Proesmans, and L. Van Gool, "Instance Segmentation by Jointly Optimizing Spatial Embeddings and Clustering Bandwidth," arXiv:1906.11109v2, 2019.

[76] S. Kong and C. Fowlkes, "Recurrent Pixel Embedding for Instance Grouping," arXiv:1712.08273v1, 2017.

[77] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," arXiv:1803.01534v4, 2018.

[78] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask Scoring R-CNN," arXiv:1903.00241v1, 2019.

[79] K. Chen *et al.*, "Hybrid Task Cascade for Instance Segmentation," arXiv:1901.07518v2, 2019.

[80] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "DetNet: Design Backbone for Object Detection," ed. Cham: Cham: Springer International Publishing, 2018, pp. 339-354.

[81] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and A. Hartwig, "MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features," *arXiv.org,* 2017.

[82] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," arXiv:1712.00726v1, 2017.

[83] Y. Lee and J. Park, "CenterMask : Real-Time Anchor-Free Instance Segmentation," arXiv:1911.06667v6, 2019.

[84] Y. Wen, F. Hu, J. Ren, X. Shang, L. Li, and X. Xi, "Joint multi-task cascade for instance segmentation," *Journal of real-time image processing,* vol. 17, no. 6, pp. 1983-1989, 2020, doi: 10.1007/s11554-020-01007-5.

[85] X. Zhang, H. Li, F. Meng, Z. Song, and L. Xu, "Segmenting Beyond the Bounding Box for Instance Segmentation," *IEEE transactions on circuits and systems for video technology,* pp. 1-1, 2021, doi:

10.1109/TCSVT.2021.3063377.

[86] Q. Wen *et al.*, "Automatic Building Extraction from Google Earth Images under Complex Backgrounds Based on Deep Instance Segmentation Network," *Sensors (Basel),* vol. 19, no. 2, p. 333, 2019, doi: 10.3390/s19020333.

[87] Z. Xu, S. Liu, J. Shi, and C. Lu, "Outdoor RGBD Instance Segmentation With Residual Regretting Learning," *IEEE Trans Image Process,* vol. 29, pp. 5301-5309, 2020, doi: 10.1109/TIP.2020.2975711.

[88] B. Zhang and J. Zhang, "A Traffic Surveillance System for Obtaining Comprehensive Information of the Passing Vehicles Based on Instance Segmentation," *IEEE transactions on intelligent transportation systems,* pp. 1-16, 2020, doi: 10.1109/TITS.2020.3001154.

[89] J. Chen, G. Wang, L. Luo, W. Gong, and Z. Cheng, "Building Area Estimation in Drone Aerial Images Based on Mask R-CNN," *IEEE geoscience and remote sensing letters,* vol. 18, no. 5, pp. 891-894, 2021, doi: 10.1109/LGRS.2020.2988326.

[90] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Microsoft Research,* 2015.

[91] G. Huang, Z. Liu, and K. Weinberger, "Densely Connected Convolutional Networks," *arXiv.org,* 2018.

[92] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84-90, 2017, doi: 10.1145/3065386.

[93] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv.org,* 2015.

[94] C. Szegedy *et al.*, "Going Deeper with Convolutions," arXiv:1409.4842v1, 2014.

[95] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," arXiv:1611.05431v2, 2016.

[96] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep Layer Aggregation," arXiv:1707.06484v3, 2017.

[97] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," *arXiv.org,* 2016.

[98] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," arXiv:1512.00567v3, 2015.

[99] Y. Chen, J. Li, H. Xiao, X. Jin, and J. Feng, "Dual Path Networks," *arXiv.org,* 2017.

[100] W. Wang, X. Li, J. Yang, and T. Lu, "Mixed Link Networks," arXiv:1802.01808v1, 2018.

[101] F. Guo, Y. Qian, Y. Wu, Z. Leng, and H. Yu, "Automatic railroad track components inspection using real-time instance segmentation," *Computer-aided civil and infrastructure engineering,* vol. 36, no. 3, pp. 362-377, 2021, doi: 10.1111/mice.12625.

[102] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A New Multi-Scale Backbone Architecture," *IEEE Trans Pattern Anal Mach Intell,* vol. 43, no. 2, pp. 652-662, 2021, doi: 10.1109/TPAMI.2019.2938758.

[103] L. Tsung-Yi, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *arXiv.org,* 2017.

[104] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," vol. 2016-, ed, 2016, pp. 779-788.

[105] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "DetNet: A Backbone network for Object Detection," arXiv:1804.06215v2, 2018.

[106] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861v1, 2017.

[107] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," arXiv:1801.04381v4, 2018.

[108] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," arXiv:1707.01083v2, 2017.

[109] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," arXiv:1602.07360v4, 2016.

[110] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," arXiv:1610.02357v3, 2016.

[111] C.-Y. Wang, H.-Y. M. Liao, I. H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," arXiv:1911.11929v1, 2019.

[112] K. K. Pal and S. K. S., "Preprocessing for image classification by convolutional neural networks," *IEEE International Conference On Recent Trends In Electronics Information Communication Technology,* pp. 1778-1781, May 20-21 2016.

[113] J. Yim and K.-A. Sohn, "Enhancing the Performance of Convolutional Neural Networks on Quality Degraded Datasets," arXiv:1710.06805v1, 2017.

[114] J. Qin, Y. Zhang, H. Zhou, F. Yu, B. Sun, and Q. Wang, "Protein Crystal Instance Segmentation Based on Mask R-CNN," *Crystals (Basel),* vol. 11, no. 2, p. 157, 2021, doi: 10.3390/cryst11020157.

[115] G. F. C. Campos, S. M. Mastelini, G. J. Aguiar, R. G. Mantovani, L. F. d. Melo, and S. Barbon Jr, "Machine learning hyperparameter selection for Contrast Limited Adaptive Histogram Equalization," *EURASIP journal on image and video processing,* vol. 2019, no. 1, pp. 1-18, 2019, doi: 10.1186/s13640-019-0445-4.

[116] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *arXiv.org,* 2017.

[117] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks," *arXiv.org,* 2016.

[118] K.-H. Kim, S. Hong, B. Roh, Y. Cheon, and M. Park, "PVANET: Deep but Lightweight Neural Networks for Real-time Object Detection," *arXiv.org,* 2016.

[119] T. Nakazawa and D. V. Kulkarni, "Wafer Map Defect Pattern Classification and Image Retrieval Using Convolutional Neural Network," *IEEE Transactions on Semiconductor Manufacturing,* vol. 31, no. 2, pp. 309-314, 2018, doi: 10.1109/tsm.2018.2795466.

[120] Y. Liu, F. Yang, and P. Hu, "Small-Object Detection in UAV-Captured Images via Multi-Branch Parallel Feature Pyramid Networks," *IEEE access,* vol. 8, pp. 145740-145750, 2020, doi: 10.1109/ACCESS.2020.3014910.

[121] Z. Li and F. Zhou, "FSSD: Feature Fusion Single Shot Multibox Detector," arXiv:1712.00960v3, 2017.

[122] L. Taylor and G. Nitschke, "Improving Deep Learning using Generic Data Augmentation," *arXiv.org,* 2017.

[123] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems With Applications,* vol. 95, pp. 43-56, 2018, doi: 10.1016/j.eswa.2017.11.028.

[124] R. Takahashi, T. Matsubara, and K. Uehara, "Data Augmentation using Random Image Cropping and Patching for Deep CNNs," *IEEE Transactions on Circuits and Systems for Video Technology,* pp. 1-1, 2019, doi: 10.1109/TCSVT.2019.2935128.

[125] H. Huang, H. Zhou, X. Yang, L. Zhang, L. Qi, and A.-Y. Zang, "Faster

R-CNN for marine organisms detection and recognition using data augmentation," *Neurocomputing,* vol. 337, pp. 372-384, 2019, doi: 10.1016/j.neucom.2019.01.084.

[126] M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, and S. W. Baik, "Multi-grade brain tumor classification using deep CNN with extensive data augmentation," *Journal of Computational Science,* vol. 30, pp. 174-182, 2019, doi: 10.1016/j.jocs.2018.12.003.

[127] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, no. 1, pp. 1-48, 2019, doi: 10.1186/s40537-019-0197-0.

[128] M. Everingham *et al.*, "The Pascal Visual Object Classes (VOC) Challenge," *International journal of computer vision,* vol. 88, no. 2, pp. 303-338, 2010, doi: 10.1007/s11263-009-0275-4.

[129] M. Everingham *et al.*, "The Pascal Visual Object Classes Challenge: A Retrospective," *International journal of computer vision,* vol. 111, no. 1, pp. 98-136, 2015, doi: 10.1007/s11263-014-0733-5.

[130] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue, "Object Detection from Scratch with Deep Supervision," arXiv:1809.09294v2, 2018.

[131] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation," *arXiv.org,* 2017.

[132] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *arXiv.org,*

2015.

[133] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," vol. 15, ed, 2011, pp. 315-323.

[134] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998, doi: 10.1109/5.726791.

[135] V. Santhanam and L. S. Davis, "A Generic Improvement to Deep Residual Networks Based on Gradient Flow," *IEEE Transactions on Neural Networks and Learning Systems,* pp. 1-10, 2019, doi: 10.1109/TNNLS.2019.2929198.

[136] A. L. KABADE and D. V. G. Sangam, "Canny edge detection algorithm," *International Journal of Advanced Research in Electronics and Communication Engineering,* vol. 5, no. 5, May 2016.

[137] Ş. Öztürk and B. Akdemir, "Comparison of Edge Detection Algorithms for Texture Analysis on Glass Production," *Procedia, social and behavioral sciences,* vol. 195, pp. 2675-2682, 2015, doi: 10.1016/j.sbspro.2015.06.477.

[138] O. R. Vincent and O. Folorunso, "A Descriptive Algorithm for Sobel Image Edge Detection," Proceedings of Informing Science & IT Education Conference (InSITE) 2009.

[139] C. Cortes, M. Mohri, and A. Rostamizadeh, "L2 Regularization for Learning Kernels," arXiv:1205.2653v1, 2012.

[140] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers:

Surpassing Human-Level Performance on ImageNet Classification," arXiv:1502.01852v1, 2015.

[141] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv.org,* 2016.

[142] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," *International journal of computer vision,* vol. 128, no. 2, pp. 336-359, 2020, doi: 10.1007/s11263-019-01228-7.

초록

반도체 제조에서 결함 검출은 높은 수율을 유지하는데 중요합니다. 전형적으로, 반도체 웨이퍼의 결함은 제조 공정에서 발생하고 있습니다. 반도체 포토리소그래피 공정 검사에 사용되는 대부분의 컴퓨터 비전 시스템들은 여전히 외부 환경 변화에 민감한 이미지 처리 알고리즘을 사용하고 있어서 검사 오류가 자주 발생하고 있습니다. 따라서, 이미지 처리 알고리즘의 장점과 딥 러닝의 장점을 융합하여 이 문제를 해결하려고 합니다.

이 논문에서 우리는 반도체 포토리소그래피 검사와 같이 훈련 데이터 세트가 제한된 상황에서 향상된 기능 맵을 추출하기 위해 이미지 분할 검출기(Image Segmentation Detector, 이하 ISD)를 제안합니다. ISD는 이미지 분할을 위한 최신 Mask R-CNN 프레임 워크의 새로운 백본 네트워크로 사용합니다. ISD는 4 개의 조밀한 블록과 4 개의 전환 레이어로 구성합니다. 특히, ISD의 각 조밀한 블록은 보다 컴팩트함을 위해 단축 연결 및 동적 성장률을 가지고 레이어에서 생성된 피쳐 맵을 결합하고 있습니다. ISD는 최근 적용하고 있는 전이 학습 방법을 사용하지 않고 처음부터 훈련합니다. 또한, ISD는 합성곱 신경망(Convolutional Neural Network, 이하 CNN)의 향상된 기능 맵을 추출하기 위해 우리가 설계한 이미지 필터를 통해 사전 처리된 이미지 데이터 세트로 훈련을 합니다. ISD의 설계 핵심 원칙 중 하나는

소형화로 실시간 문제를 해결하고 리소스에 제한이 있는 장치에 적용하는데 중요한 역할을 하게 합니다.

모델을 실증적으로 입증하기 위해 이 논문에서는 현재 운영 중인 반도체 제조 장비에 내장된 컴퓨터 비전 시스템에서 획득한 실제 이미지를 사용합니다. ISD는 가장 일반적인 성능 측정 지표인 평균 정밀도에서 최첨단 백본 네트워크 보다 일관되게 더 나은 성능을 얻습니다. 특히, ISD는 베이스 라인으로 삼은 DenseNet 보다 파라미터들이 4배 더 적지만, 성능이 우수 합니다. 우리는 또한 ISD가 Mask R-CNN 백본 네트워크로 주로 사용하는 ResNet 보다 268배 훨씬 더 적은 파라미터들을 가지고, 추가 데이터 또는 사전 훈련된 모델을 사용하지 않고, 성능에서 비슷하거나 더 나은 결과를 얻을 수 있음을 관찰합니다. 우리의 실험 결과들은 ISD가 제한된 훈련 데이터 세트만으로 실시간 및 우수한 성능을 요구하는 반도체 산업의 다양한 분야들에서 많은 미래의 이미지 분할 연구 노력에 유용할 수 있음을 보여줍니다.