UNSUPERVISED AND SEMI-SUPERVISED LEARNING IN AUTOMATIC INDUSTRIAL IMAGE INSPECTION

by

Weitao Tang

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Department of Computer and Information Technology

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Baijian Yang, Chair

Department of Computer and Information Technology

Dr. Tonglin Zhang

Department of Statistic

Dr. Milan Rakita

Department of Engineering Technology

Dr. Wenhai Sun

Department of Computer and Information Technology

Approved by:

Dr. Kathryne A Newton

To my parents, professors, and friends

who helped and comfort me during my theatricality Ph.D. career.

ACKNOWLEDGMENTS

I wish to gratefully acknowledge my thesis committee for their insightful comments and guidance, as well as my family and friends for their support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AE	autoencoder
CAE	convolutional autoencoder
CAM	class activation map
CNN	convolutional neural network
DAE	denoised autoencoder
GAN	generative adversarial network
NN	neural network
OC-SVM	one-class support vector machines

GLOSSARY

Supervised Machine Learning – It is a category of machine learning algorithms that require all the training data to be annotated. The algorithm can only learn from annotated data.

Semi-supervised Machine Learning – It is a category of machine learning algorithms which can learn from a partially annotated dataset. The algorithm can learn from both annotated and unannotated data at the same time.

Unsupervised Machine Learning – It is a category of machine learning algorithms which can learn from data without any annotation. The algorithm learns only from unannotated data.

Unsupervised Anomaly Detection – It is the task of identifying those data samples that deviate from common behaviors. Different from supervised classification, which requires data for all categories, anomaly algorithm requires only good images.

Weakly-supervised Learning – It is a category of machine learning algorithms that require less detailed annotation for its targeting task than the corresponding methods of supervised learning. For instance, an algorithm that can locate the defect in the image precisely with only the annotations of good or defective images.

Clustering - It is the task of splitting a given data set into several groups without any annotation. Each group of data is called a cluster.

ABSTRACT

It has been widely studied in industry production environment to apply computer version on X-ray images for automatic visual inspection. Traditional methods embrace image processing techniques and require custom design for each product. Although the accuracy of this approach varies, it often fall short to meet the expectations in the production environment. Recently, deep learning algorithms have significantly promoted the capability of computer vision in various tasks and provided new prospects for the automatic inspection system. Numerous studies applied supervised deep learning to inspect industrial images and reported promising results. However, the methods used in these studies are often supervised, which requires heavy manual annotation. It is therefore not realistic in many manufacturing scenarios because products are constantly updated. Data collection, annotation and algorithm training can only be performed after the completion of the manufacturing process, causing a significant delay in training the models and establishing the inspection system. This research was aimed to tackle the problem using unsupervised and semi-supervised methods so that these computer vision-based machine learning approaches can be rapidly deployed in real-life scenarios. More specifically, this dissertation proposed an unsupervised approach and a semi-supervised deep learning method to identify defective products from industrial inspection images. The proposed methods were evaluated on several open source inspection datasets and a dataset of X-Ray images obtained from a die casting plant. The results demonstrated that the proposed approach achieved better results than other state-of-the-art techniques on several occasions.

CHAPTER 1. INTRODUCTION

This chapter serves as an overview of the entire research. It first presents the background of the target domain and then introduces the research questions. It is followed by the discussions on the significance, assumptions, limitations, and delimitations of this research.

1.1 Background

Visual inspection is an essential step in quality control in modern factories. The inspection tasks of many products like X-Ray scanning of die casting can be extremely complicated. Human inspectors are the dominant forces in the inspection tasks due to high processing and understanding ability of human vision. However, humans are not machines and suffer various problems like eye fatigue, mental exhaustion, and individual bias. Kang, Ramzan, Sarkar, and Imran (2018) pointed out that the managing of human inspectors can be complicated. Many factors, such as location, working time interval, the number of inspectors, and inspection strategies, need to be taken into account for a systematic design. Therefore, the accurate and efficient computer-based automatic vision inspection systems are desired to assist or replace human vision inspection.

Such automatic vision inspection systems have been researched for decades (Thomas, Rodd, Holt, & Neill, 1995). However, it is still challenging to apply them in many complex scenarios. A typical automatic industrial image inspection system consists of several components like image acquisition, automatic inspection, light-adjusting system, and others. Among them, the inspection algorithm is the most essential component since all other parts revolve around it. Traditionally, such inspection algorithms are developed using image processing technique. These image processing techniques can only capture the low-level information about images and often require handcraft design according to the specific product (Demant, Garnica, & Streicher-Abel, 2013). Therefore, they often fail to achieve decent performance in complex inspection scenarios. In addition, these techniques need to be explicitly designed even for similar tasks and thus have poor generalizability.

In the current decade, deep learning algorithms have exhibited a significant improvement of performance in all vision tasks. Hinton, Osindero, and Teh (2006) introduced a brief deep network and made the “age of deep learning” thrive. Krizhevsky, Sutskever, and Hinton (2012) proposed the AlexNet, a convolutional neural network (CNN) intended for image classification. It achieved a Top-5 error rate (rate of ground truth not in the top 5 predictions) of 15.3% in the ImageNet competition of image classification (Deng et al., 2009), which surpassed the next best result by more than 10%. Since then, the CNN based algorithms dominated many tasks, including but not limited to classification, object detection, tracking, 3D reconstruction, image registration, and segmentation, in popular computer vision competition datasets, such as ImageNet (Deng et al., 2009), Common Objects in Context (COCO) (T.-Y. Lin et al., 2014), and PASCAL Visual Object Classes (PASCAL VOC) (Everingham, Van Gool, Williams, Winn, & Zisserman, 2010). The development of deep learning has provided new opportunities for automatic vision inspection.

In the most recent five years, various studies applied deep learning methods like classification, object detection, segmentation, and others in inspection scenarios and produced satisfactory performance. These studies cover diverse scenarios including steel plates, LED chips, aircraft engine blades, casting products, and so on. However, the foundation of most of these studies, the labeling effort, is often overlooked. Most of these studies applied supervised deep learning algorithms, which require labels for every image in the dataset. It is a routine that thousands or more labeled images are required. In general, more data brings better performance. This causes a high consumption of both labor and time.

Different from supervised learning methods, semi-supervised learning and unsupervised learning require less or even no annotations. Semi-supervised classification, objection detection, and segmentation, as well as unsupervised clustering and anomaly detection, hold a great potential in automatic industrial image inspection. Once implemented correctly, the semi-supervised methods can achieve similar performance to their corresponding supervised methods while using fewer annotations. As for unsupervised learning, the anomaly detection requires only the images without defects to train and it detects the images with defects as anomaly. Meanwhile, the unsupervised clustering algorithms automatically split data into several categories without any annotation. Although clustering cannot match supervised classification in terms of accuracy, it performs well in analyzing the composition of existing data. Moreover, clustering result can be further corrected by human annotators to provide annotation for supervised training. Such process can be more convenient compared with annotating from scratch. In this research, the application of unsupervised learning and semi-supervised learning in automatic industrial image inspection was investigated. The proposed methods were tested on various open source datasets as well as a dataset provided by a manufacturer.

1.2 Research Question

The research questions were as follows:

- Can the unsupervised anomaly detection be applied in automatic industrial image inspection?
- Can the semi-supervised learning be applied in automatic industrial image inspection?

1.3 Significance

The current human vision inspection system is expensive and not reliable. Human inspectors often need specialized training to acquire inspection skills. Eye fatigue is also unavoidable since inspection jobs are repetitive and tiresome. Another problem is that although the human vision is highly effective, it is often hard to define a clear standard threshold for judgment, and the quality of products is sometimes uniform. For instance, Kang et al. (2018) described the inspection scenario of the blisters in cooper casting parts, there was not strict metric to describe the blisters, and different inspectors may applied different standards.

Meanwhile, the current studies of the automatic inspection system often apply supervised deep learning methods that require tremendous annotation effort. Deep learning algorithms are known for data hunger (Goodfellow, Bengio, & Courville, 2016). Training an inspection algorithm with decent performance often requires thousands of image or more. Annotating such huge numbers of images for classification task already consumes a large amount of labor resources. Moreover, more complicated tasks like object detection and segmentation require far more efforts in annotation compared with classification tasks. This limits their application in many manufacturing industries, especially those with complicated and diverse products. In some scenarios, the products can be constantly updated. The time period and cost of image collection, annotation, and algorithm design are subject to minimization. Another problem is that deep learning algorithms often favor unbiased dataset, meaning the number of images in different categories is supposed to be balanced. However, a mature manufacturing process creates far more defect-free images than defective images, which makes collecting large number of defective images difficult. In some cases, the unprecedented defects that do not exist in training data can also occur. Conventional supervised classification algorithms can not handle such case and cause failure. Therefore, the methods that require fewer annotations and hold greater robustness are highly desired.

In recent years, the performance of semi-supervised methods and unsupervised methods has improved significantly. The semi-supervised methods are capable of performing the same task as their corresponding supervised methods while requiring fewer labels. These methods learn from datasets with only a small portion of data annotated. This significantly reduces the amount of effort required for data annotation, which is the main burden when preparing the dataset.

Meanwhile, the unsupervised methods are capable of learning information from the data without any supervision. For instance, unsupervised anomaly detection only requires good images for training. It learns the characteristics of non-defective images and spots defective images as anomalies since defective images do not share the same characteristics as good images. Despite no defective images required for training, another advantage of unsupervised anomaly detection is that they are capable to detect unprecedented defects while supervised classification methods often fail in this respect.

Unsupervised clustering algorithms are also promising in industrial images related tasks. They can be applied to the data stored from previous inspection process to analyze the composition of past failures. Such information can help the manufacturing engineers to identify the causes and ratios of different types of defects, which benefits the further optimization of the manufacturing process. It can also serve as a starting point for human annotation. Correcting a small number of false predictions in the clustering result can be far easier than annotating every single image from scratch. This provides a great foundation for training supervised and semi-supervised algorithms.

Despite various benefits of semi-supervised and unsupervised methods as mentioned above, there are only few studies that discussed their application to industrial images. Therefore, such research is highly desired and of significance.

1.4 Assumptions

The assumptions for this research included:

- The performance of the proposed methods on the datasets applied in this research represented their performance in real-world inspection scenarios.
- Results from the testing datasets were representative.

1.5 Limitations

The limitations of this research included:

- This research only focused on the algorithm design component of the whole automatic visual inspection system since other components like algorithm deployment and hardware setting required multiple discipline corporations.
- There are numerous industrial inspection scenarios and this research could not cover all of them.

1.6 Delimitations

The delimitations of this research included:

- An automatic vision inspection system contains many other parts except for the inspection algorithm. However, they were not studied in this research.
- The proposed methods only covered unsupervised anomaly detection and semi-supervised classification. Other directions such as unsupervised clustering, semi-supervised object detection and semi-supervised segmentation were not investigated in this research.

1.7 Summary

The first chapter introduces the scope and significance of the research. It also enumerates the questions, assumptions, limitations, and delimitations of it.

CHAPTER 2. REVIEW OF LITERATURE

An automatic industrial image inspection system consists of different components ranging from hardware to software. This research focused on the deep learning based inspection algorithms. This section reviews the research of applying deep learning algorithms in industrial image inspection scenarios, including both supervised learning and unsupervised learning. It also reviews the research of semi-supervised classification and contrastive learning.

2.1 Supervised Learning in Industrial Image Inspection

2.1.1 Classification

Classification algorithms have been widely applied in industrial image inspection scenarios by classifying images into several types. A fully annotated dataset with thousands or more images is often required. Masci, Meier, Ciresan, Schmidhuber, and Fricout (2012) applied (CNN) to classify different types of defects on steel plates. A significant improvement on the traditional image processing based approaches was reported. It should be noted that their dataset contained only the images of different defects without the images of good products, so that their algorithm only distinguished between different types of defects. Kuo, Hsu, Liu, and Wu (2014) classified LED chip images. The features of the images were extracted using classic hand-crafted approaches and treated as the input of a two-layer fully connected neural network (NN). The manually crafted feature extractors were weak compared to deep learning based feature extraction, and customization was required for each type of LED chips. Constrained by the poor ability of the hand-crafted feature extraction, the performance was limited. Differently, P.-H. Chen and Ho (2016) utilized Overfeat, a powerful pretrained CNN network, as features extractors on steel surface defects. Then, a simple SVM classifier was trained on the extracted features. Their algorithm showed high performance on simple texture patterns defects but struggled on more complex defects. Additionally, another hand-crafted operator was introduced to solve this issue. Similar to Masci et al. (2012), this work was more concerned about classifying defective products into different categories rather than detecting defective products from all the products. Another limitation of P.-H. Chen and Ho (2016); Kuo et al. (2014); Masci et al. (2012) was that the defects were not located in their studies. In addition to Masci et al. (2012) and P.-H. Chen and Ho (2016), Yi, Li, and Jiang (2017) investigated different data augmentation techniques to mine the information within the limited dataset. The impact of each data augmentation techniques was studied individually. A combination of all the augmentation techniques remarkably leveraged the performance of the CNN. However, the trained algorithm was incapable to identify defects either due to the lack of good images in dataset.

With the extra designs like sliding window or class activation map (CAM), classification algorithms can be extended to coarse localization. Similar to Kuo et al. (2014), H. Lin, Li, Wang, Shu, and Niu (2019) also targeted LED chip inspection with a full deep learning classification pipeline, from feature extraction to feature classification. Different from Kuo et al. (2014), their approach achieved a satisfactory performance. They also employed a CAM and the gradient of neural network to perform coarse localization for the defect. The classification and coarse localization were achieved in a single CNN model. However, their algorithm was trained on a fairly large dataset with $30,000$ images of a specific type of LED chip. This suggested that a fairly considerable amount of time and resources were spent on data collection and annotation. The algorithm could only be applied to the single type of LED chip on which it was trained. Apparently, the supervised classification approach is data hungry and labor intensive. Mery (2020) targeted metal casting inspection. They applied CNN classification to the small patches of a whole image to identify whether defect exists in the patch. By covering the whole image with small patches through a sliding window approach, coarse localization was achieve as well. GAN (Goodfellow et al., 2014) was introduced to generate negative samples and to enlarge the training set.

2.1.2 Segmentation

Different from those classification algorithms which output the category of an image, the segmentation algorithms output a category for each pixel in the image. Therefore, they provide more information on the defect, including size, shape, and position. However, these algorithms are trained with pixel-wise annotation, which consumes significantly more time and labor. Bian, Lim, and Zhou (2016) applied a multi-scale fully convolutional network to evaluate the defects on the surface of aircraft engine blades. Unlike normal inspection scenarios, annotating defect on aircraft engine blades required considerable professional training and experience, so that the amount of available data was limited. To reduce the overfitting caused by insufficient dataset, the segmentation network was trained at different scales and then combined together. Their approach achieved the comparable performance to human vision, and outperformed one of three annotators. The large amount of time and labor is acceptable in high precision scenarios like aircraft engine blades. However, this is generally not acceptable for many other industries. Racki, Tomazevic, and Skocaj (2018) designed a network structure that enabled the classification and segmentation of surface defects at the same time. The classification part took the segmentation feature map as input. Similar to Bian et al. (2016), their algorithm required pixel-wise annotation. Tabernik, Šela, Skvarč, and Skočaj (2020) modified the model architecture from Racki et al. (2018) to increase the receptive field and improve the capability of capturing fine feature details.

2.2 Weakly-supervised and unsupervised anomaly Detection

Compared with supervised learning methods, weakly-supervised and unsupervised learning methods require less annotation for training. More specifically, weakly supervised algorithm can be trained with simpler annotations to perform more complex tasks, for instance, using image level annotation to obtain pixel level prediction. Meanwhile, unsupervised anomaly detection algorithms require only the annotation of good images for training. In 2013, before the deep learning era, Benmoussat, Guillaume, Caulier, and Spinnler (2013) designed an algorithm to detect anomaly on thermographic images. A multivariate Gaussian model was constructed through the combination of singular values decomposition (SVD), maximum noise fraction (MNF), and hyper-spectral signal identification by minimum error (HySime). More recently, Jin et al. (2019) applied convolutional autoencoder to (CAE) to detect anomaly during the addictive manufacturing process. Instead of inspecting the finished product, the algorithm monitored the printing stage when the material is accumulated together. Lehr, Sargsyan, Pape, Philipps, and Krüger (2020) also studied the application of CAE in the anomaly detection of industrial images.

However, their dataset was fully synthetic, which caused a lack of the variation of position, angles, light condition, and others. Except the CAE, no additional strategies were implemented to reduce the impact of product placements and noise during image acquisition.

A number of researches studied the anomaly detection of surface defect detection scenarios, where the products are often plates or sheets, and have no geometries. The inspection in such scenario mainly focuses on texture information instead of geometric information. These products are often mass produced, semi-finished products like wood plates or steel plates, and will be further processed into the desired geometry. Two public datasets, Silvén, Niskanen, and Kauppinen (2003) and Oliveira and Correia (2014), which contain tens of thousands of images, are commonly used in these researches. Collecting the data of such scale is costly and unrealistic in various industries. Kimura and Yanagihara (2018) utilized the generative adversarial networks (GAN) from Radford, Metz, and Chintala (2015) to detect surface anomaly. The discriminator of the GAN served as the classifier and the surface defect was localized by its response map. Staar, Lütjen, and Freitag (2019) adopted a triplet network to detect anomaly on a public surface defect dataset (Wieler & Hahn, 2007). However, training a triplet network required the image-level annotation of several different types of categories, so that their methods were weakly supervised instead of being unsupervised. Li, Xu, Gao, Wang, and Shao (2020) studied CAE anomaly detection as Lehr et al. (2020) on real world datasets. Different from Lehr et al. (2020), they introduced two additional discriminators to compete with the CAE, which leveraged the reconstruction performance and reduced the effect of image acquisition noises. A downside was that the extra classifier required a number of defective images for training, which made their algorithm weakly supervised instead of being unsupervised compared with Lehr et al. (2020). Liu, Li, Wen, Chen, and Yang (2019) combined a GAN network with a classic algorithm one-class SVM (OC-SVM) to detect defects on steel surface. The features extracted by GAN served as the input of the OC-SVM. Hu et al. (2020) also applied GAN to detect anomaly on cloth surface. However, GAN algorithms are known to be tricky to train, and they often fail without any specific reason. This is acceptable for mass produced, semi-finished products like steel sheets, wood boards, and fabrics, but not for other constantly updated and less mass produced products.

2.3 Semi-supervised Image Classification

Semi-supervised image classification has long been studied. In such settings, only a small portion of dataset is annotated. Their performances are often slightly inferior to the supervised learning with all data annotated, but superior to the supervised learning with same degree of annotation. Erhan, Courville, Bengio, and Vincent (2010) trained the network in two stages. The network was first trained with an unsupervised task, and the dimension was reduced by an autoencoder (AE), with all available data used. In the second stage, the network was fine-tuned using annotated data. Lee et al. (2013) proposed to use the maximum prediction of unannotated images as their pseudo-label and they were trained with a supervised loss in the fine-tuning stage. Denoised autoencoder (DAE) and dropout were introduced to boost the generalizability and robustness of the network. Their methods maximized the predicted probability of unannotated images, thus reducing the prediction entropy. The pseudo-labeling became a commonly used method subsequently. Rasmus, Valpola, Honkala, Berglund, and Raiko (2015) introduced ladder network Valpola (2015) in semi-supervised classification. The ladder network resembled the DAE. However, instead of just introducing noises into the bottle neck layer, noises were introduced into every layer of the encoder. Such design increased the robustness of the network. The supervised classification of annotated data and unsupervised reconstruction of all data were trained collectively.

More recent works broke the boundary between the unsupervised task and supervised task, which caused the structures like AE to be abandoned. Sajjadi, Javanmardi, and Tasdizen (2016) inputted all the images into the network with multiple passes while applying random data augmentation, dropout, and random max-pooling. A consistency loss was proposed for the output of different passes of the same image. To reduce the prediction entropy of the output of unannotated images, a mutual exclusivity loss was proposed as well. Laine and Aila (2016) ensembled the predictions of unknown images from different epochs as the pseudo targets to train the current network. They also implemented a consistency loss for different image augmentations and different network dropouts. Laine and Aila (2016) significantly leveraged the performance of semi-supervised classification. Despite its success, the pseudo targets were only updated in every epoch, which reduced its performance on large datasets since updates could not keep up with neural network training. Tarvainen and Valpola (2017) proposed a mean teacher policy, which averaged the model weights from different steps instead of using the model predictions from different epochs in Laine and Aila (2016). Combined with residual networks, He, Zhang, Ren, and Sun (2016), the performance of semi-supervised classification was further improved to a significant extent. Xie, Dai, Hovy, Luong, and Le (2020) proposed a combination of entropy minimization and pseudo labeling to achieve a remarkable performance. In terms of pseudo labeling, they applied confidence threshold and entropy sharpening. They also conducted a comprehensive study on the impact of different augmentations. Sohn et al. (2020) adopted the same strategy as Xie, Dai, et al. (2020), except that they removed entropy sharpening. AutoAugment (Cubuk, Zoph, Mane, Vasudevan, & Le, 2019) was applied to search for the better augmentation policy. Xie, Luong, Hovy, and Le (2020) targeted a situation where a large portion of the unlabeled images are from other related categories instead of the target categories. They iterated a two-step training loop. Initially, a teacher model was trained with annotated images. Secondly, the teacher model generated pseudo labels on unlabeled images. Thirdly, a student model was trained with both annotated images and pseudo labeled images. The student model was then treated as the teacher to generate pseudo labels in step two. The noises such as dropout, stochastic model depth, and data augmentations were applied to train the student model to force it into learning from much harder scenarios than its teacher.

Despite the success of these semi-supervised classification methods, an essential problem that limits their practical applications remains less discussed. Unlike those standard research datasets like CIFAR (Krizhevsky, Hinton, et al., 2009), or ImageNet (Deng et al., 2009), all of which have an uniform number of images for each category, real-world dataset can be highly biased. This is rather significant for industrial images, since all manufacturing processes are optimized to produce fine products instead of defective ones. Good images can be dominant among the collected images. Hastily abandoning the extra good images to create a balanced training set would cause data waste and the failure to bring out the full potential of the dataset. Yang and Xu (2020) demonstrated that imbalanced learning can benefit from unannotated data through semi-supervised learning. Kim et al. (2020) proposed to gently refine the biased pseudo labels of unannotated images through convex optimization. More specifically, they assumed the same distribution between the unlabeled data and the labeled data, and then minimized the KL divergence between the pseudo distribution of the unlabeled data and the distribution of the labeled data. Hyun, Jeong, and Kwak (2020) added an regularization loss into the loss function to increase the weight of minor classes while reducing the weight of major classes. Wei, Sohn, Mellina, Yuille, and Yang (2021) proposed to use the class-rebalancing in training to adjust the sampling rate of the minor and major classes in the semi-supervised scenario. Both the reweighting regularization and class-rebalancing methods originated from the supervised learning for imbalance dataset.

2.4 Contrastive Learning

In recent years, contrastive learning has become the mainstream strategy applied to mine information from unannotated data, such as unsupervised visual representation learning (Le-Khac, Healy, & Smeaton, 2020), and unsupervised image clustering (Niu, Shan, & Wang, 2021; Van Gansbeke, Vandenhende, Georgoulis, Proesmans, & Van Gool, 2020). To train neural network without any labels, it mines information through augmentation consistency. The core idea is to perform multiple augmentations on training images and then construct both positive and negative pairs. Positive pairs consist of different augmented images from the same source image, while negative pairs consist of the augmented images from different source images. The loss function minimizes the distance between positive pairs while maximizing that between negative pairs. Figure 2.1 illustrates this process. It should be noted that only minimizing the distance between positive pairs would result in failure since this causes the network to cheat by outputting the identical vector for any input. There are noises in the unsupervised contrastive learning loss function since the augmented images whose source images are different but from the same category are also introduced into negative pairs due to a lack of labels. Another point worth mentioning is that the target is not to train a network for classification task, but to extract information from images. More specifically, the loss function is applied to the model backbone. It takes the input of vectors with a preset size, such as 512 or 1024, instead of the predicted probability of different categories. The performance is evaluated by adding a single-layer classifier onto the backbone, fine-tuning only the classifier with labeled data, and then testing the classification performance on test data. Moreover, contrastive learning training also enhances the performance of the backbone in various downstream tasks like object detection and segmentation. The knowledge learned in contrastive learning can be transferred to other tasks.

Despite its success in unsupervised learning, contrastive learning has also been applied in supervised learning. Khosla et al. (2020) applied the contrastive learning framework in supervised learning by constructing positive pairs and negative pairs through labels. They proved that the training by contrastive loss with supervision outperforms the training by conventional cross entropy loss in accuracy and parameter sensitivity. Conventional cross entropy loss assigns a centroid vector to each category and minimizes the distance between model output and the corresponding centroid vector. Despite its simplicity, a number of studies still pointed out its weaknesses including noise sensitivity (Sukhbaatar, Bruna, Paluri, Bourdev, & Fergus, 2014; Z. Zhang & Sabuncu, 2018), poor margins (Cao, Wei, Gaidon, Arechiga, & Ma, 2019), and the susceptibility to adversarial example (Nar, Ocal, Sastry, & Ramchandran, 2019). On the contrary, supervised constructive learning built the vector space without preset centroid vectors, instead, it focused on the distance between each pair of samples and thus refined the vector space.

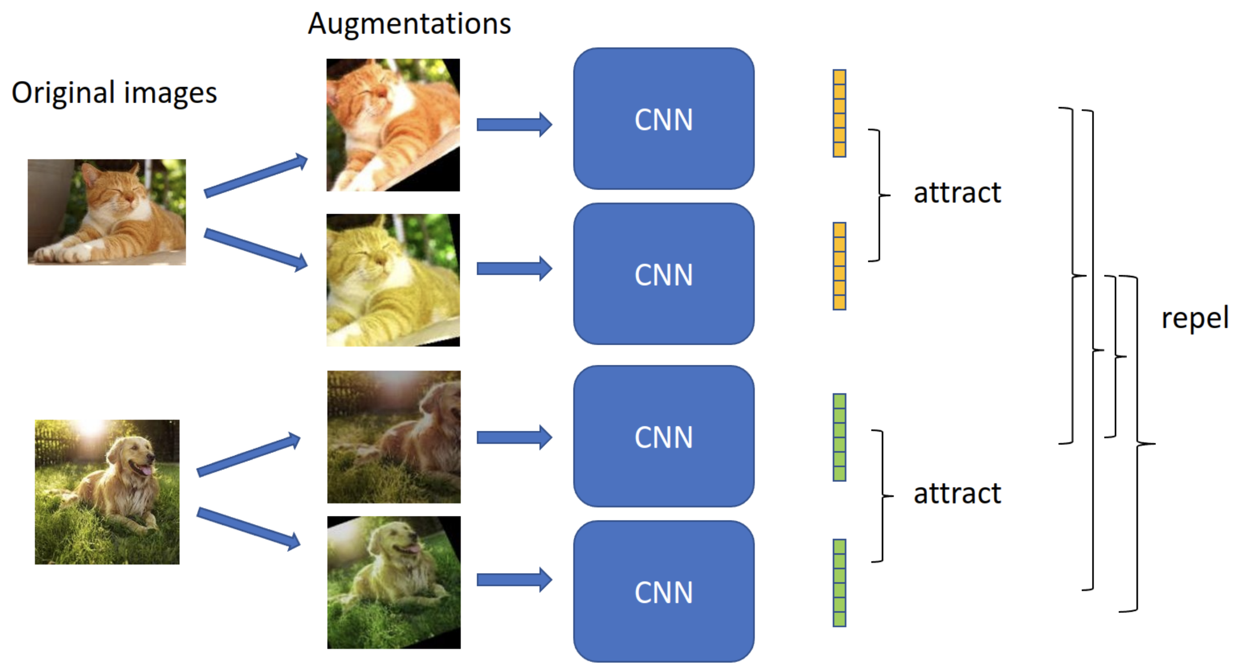


Figure 2.1. An illustration of contrastive learning

2.5 Summary

This chapter first reviews the existing research which applies supervised learning, unsupervised learning, and weakly supervised learning algorithms in industrial image inspection. The supervised learning research consists of both classification and segmentation, while the weakly supervised and unsupervised learning research often target surface defect defection. After that, the research of semi-supervised image classification and contrastive learning is also reviewed to build the methodological foundation of this research.

CHAPTER 3. METHODOLOGY

This research mainly covered two topics, the unsupervised anomaly detection and semi-supervised classification of industrial images. This chapter makes a general introduction of the methodology of each topic. Industrial images cover a wide range of scenarios and each of them requires a specific design. These detailed designs are introduced in the following chapters.

3.1 Anomaly Detection of Industrial Images

In this study, the focus was to inspect the defects with geometry instead of those on uniform surface (Tang et al., 2021). The samples of none-defect products are referred to as "good" and the samples of defective products are referred to as "defective" in anomaly detection. The algorithm was able to decide whether a product is good or defective automatically. To reduce the cost of data collection and data annotation, the algorithm was designed in a manner that only requires a small number of "good" images for training. However, different types of defects were not distinguished between in this study. The proposed method was tested on a real-world die casting inspection dataset.

3.1.1 Automatic Localization

Prior to feeding the inspection image into the algorithm, an essential process was to locate the area of interest first. This reduced the complexity of scenario, highlighted the focus of inspection, and reduced the impact of noises. The area of interest in the data were located through Hough transformations. Hough transformations are a classical category of algorithms used to locate a certain geometry in a digital image. More details about Hough transformations and their applications can be found in Leavers (1992). The flexibility of Hough transformation ensures that such localization process can be generalized to any products with different geometries. Other image processing based approaches can also be applied to locate the area of interest according the specific scenario.

3.1.2 CAE

An anomaly detection algorithm trains a model with only the good sample to learn their intrinsic characteristics. During the inference stage, the unknown sample is fed into the trained model in the same manner as training samples. The performance of the model is ideal only when the input sample shares the same characteristic as the training data, *i.e.*, and the input sample is good; otherwise, the input sample is defective. In this study, CAE was taken as the model for its high generalizability on image data.

AE is a class of unsupervised artificial neural networks. It is widely applied in various tasks including but not limited to dimension reduction, feature extraction, and data generation. AE often contains two main components, the encoder $F_w(x_i)$ and the decoder $G_{w'}(x_i)$. The encoder maps the input sample into a representation with the number of dimension far smaller than the dimension of the input data. Meanwhile, the decoder projects the dimension reduced representation back into the original input. The structure of AE can be described as a "hourglass" and the conjunction between encoder and decoder is referred to as ”bottleneck”. Such structure ensures that the representation at bottleneck preserves all the essential information about the input and the AE learns the characteristic instead of just "copying" the input to the output.

In most AEs, the consistency between input and output is ensure by a mean squared error (MSE) loss as follows:

$$L = \frac{1}{n}\sum_{i=1}^{n} ||G_{w'}(F_w(x_i)) - x_i||_2^2 \tag{3.1}$$

where n is the batch size and x_i is the ith sample within the batch.

CAE is a special type of AE designed for images. In summary, the CAE uses convolutional layers in encoder and deconvolutional layers in decoder. The convolutional layers extract information within images by sliding a weight sharing kernel across the image. More details of convolutional layers and deconvolutional layers can be found in Dumoulin and Visin (2016). Figure 3.1 presents the structure of a CAE. It should be noted that batch normalization layers and activation functions are not presented.

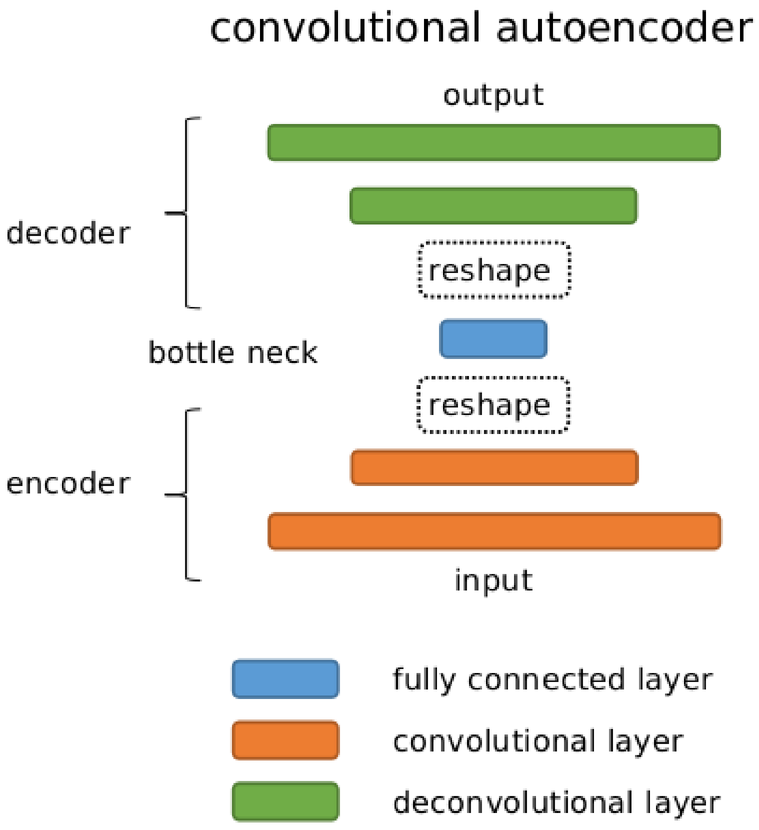


Figure 3.1. Convolutional autoencoder (CAE) structure illustration (Tang et al., 2021)

3.1.3 Inspection Strategy

The proposed inspection process was used to determine whether an image is defective or not through the reconstruction performance of trained CAE model on the inspected image. Firstly, a difference image was obtained by calculating the absolute value of the pixel-wise subtraction from the inspected image and the output of CAE. This difference image was supposed to have low values when the inspected image was good, and high values when the inspection image was defective.

An intuitive idea was to measure the average or total value of the difference image for evaluating the performance of the model. However, in the studied scenario, defect only existed in a minor region of the whole image, and was averaged out across the difference image. Moreover, there were approximately two levels of pixel values for all defects. The majority of defects had high values on the difference image and were relatively easy to inspect. However, a small portion of defects hold small values on the difference image and were easily mistaken for the model noises that were often observed at the geometry boundary.

Under the described scenario, a two-step sliding-window strategy was proposed. The first step was to identify the defects with relatively higher values, while the second step was to identify the defects with lower values and filter the model noise. The image was passed to the second step of inspection only if it succeeded in the first step of inspection; otherwise, it was directly rejected.

A sliding window which was much smaller than the original image was applied across the image. Analyzing the smaller window instead of the whole difference image reduced the averaging effect and made the defect more significant. If the average value in the sliding window exceeded a coarse threshold during sliding, the image was considered as defective.

Once the image succeeded in the first step of inspection, it was transferred to the second step of inspection. Defects remained until this step had much lower values that were likely to be mistaken for the model noise on the edge. These edge areas were removed using edge detection algorithm on the original image during the inspection. The edge detection algorithm obtained a gradient image, which had high values at the geometry edge of the original image and low values at none-edge locations. Another sliding window was applied on both the difference image and the gradient image simultaneously. The average value of the window on the gradient image was measured to exclude the impact of the model noise at the geometry edge. Once the value fell below a threshold, it was determined that the window was not at a geometry edge. This suggested that there was unlikely to be model noise in the current window. Then, a fine threshold on the averaged value of the sliding window on the difference image was calculated. To inspect those smaller or less significant defects, this fine threshold had a much lower value than the coarse threshold that defined in step one. If the average value was higher than the fine threshold, then a defect was detected in the window area, and the image was rejected as defective. If the averaged value of the window on the gradient image was higher than the gradient threshold, then the window was positioned on the geometry edge, and no inspection was performed, since there was a significant chance of model noise instead of defect. If no defect was identified after sliding the window across the whole image, then the image passes inspection and was identified as good image.

The details about the flowchart of two-step inspection are presented in Figure 3.2. There were several hyper-parameters including the window size, the gradient operator and the threshold values. These parameters depended on the characteristics of data and precision recall trade-off, and they were elaborated on in Chapter 4.

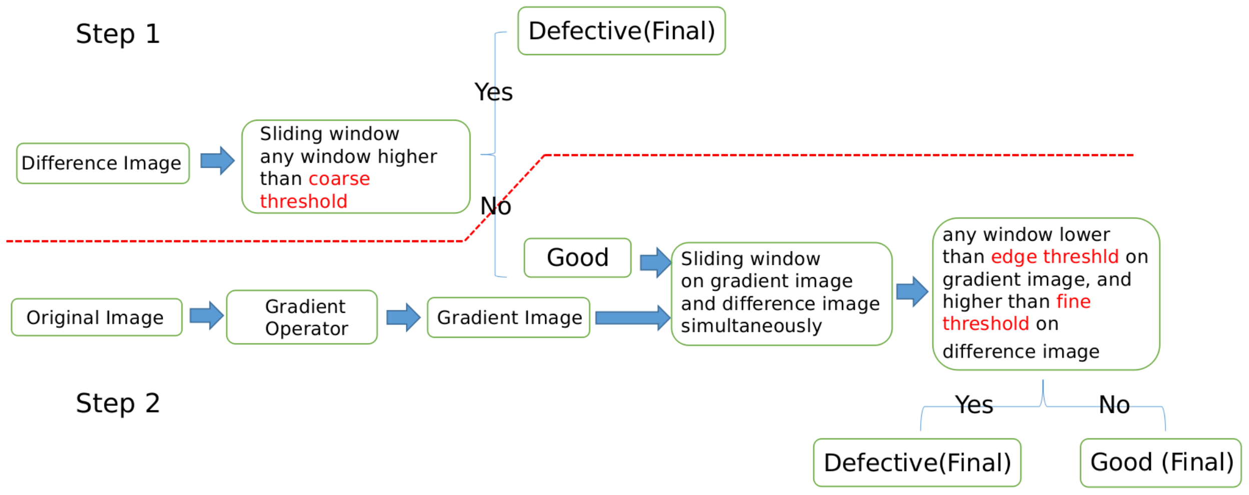

Step 1
Defective(Final)
Yes
Difference Image
Sliding window any window higher than coarse threshold
No
Good
Sliding window on gradient image and difference image simultaneously
any window lower than edge threshld on gradient image, and higher than fine threshold on difference image
Original Image
Gradient Operator
Gradient Image
Step 2
Yes
No
Defective(Final)
Good (Final)

Figure 3.2. The two-step inspection procedure pipeline (Tang et al., 2021)

3.2 Semi-supervised Annotation of Industrial Images

The unsupervised anomaly detection study distinguished good and defective product only instead of classifying different types of defects. To train a classification model that classifies different types of defects, a dataset with each types of defect is required. In reality, to collect and annotate such dataset is time-consuming and labor-intensive. In this study, the goal was to develop a method that automatically annotates a dataset with only a small amount of annotated data (10% or less). The method was implemented and tested on two open source datasets. It should be emphasized that the goal was to obtain high accuracy auto-annotations on unannotated dataset, instead of training a classifier for the K categories. The auto-annotations on unlabeled dataset were supposed to be corrected by a human annotator. After that, the classification model could be trained in conventional training, validation, and test of strategy to perform the inspection task. In real-world inspection scenarios, the whole inspection system contains much more than inspection model from hardware to software, and the performance of the model is supposed to be maximized, which requires full supervision.

As introduced in Section 2.4, contrastive learning achieved a superior performance in unsupervised visual representation learning, clustering, and supervised classification. This study further introduced the contrastive learning into semi-supervised learning. The proposed algorithm was named SemiCon (semi-supervised-contrastive-learning) since it introduced contrastive learning into semi-supervised classification.

3.2.1 Task Mathematical Specification

There were two datasets in this study: the annotated dataset and the unannotated dataset. The annotated dataset $\boldsymbol{X}$ had a total of M images from K categories and was denoted as $\boldsymbol{X} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) | i = 1 \dots M, \boldsymbol{y}_i \in (1, K)\}$, where $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ represented the annotated sample i and its category. All the $\boldsymbol{y}_i$ were known.

The unannotated dataset $\widetilde{\boldsymbol{X}}$ had a total of N images from K categories and was denoted as $\widetilde{\boldsymbol{X}} = (\widetilde{\boldsymbol{x}_i}, \widetilde{\boldsymbol{y}_i}) | i = 1 \dots N, \widetilde{\boldsymbol{y}_i} \in (1, K)\}$, where $\widetilde{\boldsymbol{x}_i}$ and $\widetilde{\boldsymbol{y}_i}$ referred to the unannotated sample i and its category. All the $\widetilde{\boldsymbol{y}_i}$ were unknown. The size of unannotated dataset $\widetilde{\boldsymbol{X}}$ was significantly larger than that of the annotated dataset $\boldsymbol{X}$. The target was to split the unannotated set of any image into K categories correctly.

3.2.2 Model Structure

The proposed model contained three modules: an encoder module, a projector module, and a classifier module.

- The encoder module $Enc()$ extracted information from the input image $\boldsymbol{x}$ into a vector $\boldsymbol{v}$ with a fixed length. The vector $\boldsymbol{v}$ was normalized into unit hypersphere so that $\boldsymbol{v} = norm(Enc(\boldsymbol{x}))$. The images from the same category were supposed to have similar vectors, while those images from different category were supposed to have different vectors. Although the $\boldsymbol{v}$ vector space was the target space to perform classification, it was not optimized directly. The vector $\boldsymbol{v}$ was forwarded to a projector module and the corresponding output was optimized. In all experiments, the architecture of the encoder was ResNet-50, and the length of vector $\boldsymbol{v}$ was 512.

- The projector module $Pro()$ took vector $\boldsymbol{v}$ and outputted another vector $\boldsymbol{z}$. The $\boldsymbol{z}$ was normalized into unit hypersphere as well, and $\boldsymbol{z} = norm(Pro(\boldsymbol{v}))$. The loss function directly optimized the $\boldsymbol{z}$ vector space, which was a function of the targeted $\boldsymbol{v}$ vector space. Such structure is often referred to as "projector head". It was first proposed in T. Chen, Kornblith, Norouzi, and Hinton (2020), and then became a conventional module in contrastive learning. The "projector head" improves the performance significantly although the underlying mechanism is not completely understood yet. The projector consisted of a fully connected (FC) layer, a relu activate function, and another FC layer. The first FC layer had an input size of 512 and an output size of 512, while the second FC layer had an input size of 512 and an output size of 128. Both FC layers have no bias.

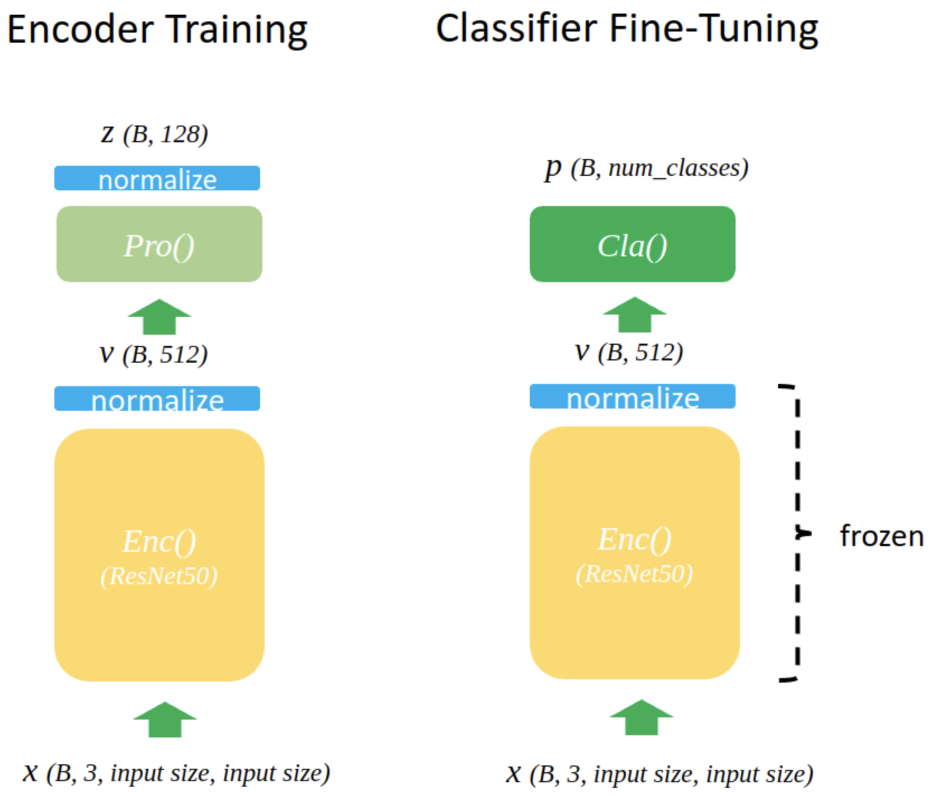


Figure 3.3. Model Modules

- The classifier module $Cla()$ mapped the target $\boldsymbol{v}$ vector into the predicted probability $\boldsymbol{p}$ of the sample $\boldsymbol{x}$. Unlike the conventional cross entropy classification, the classifier was not updated during the training of the encoder. Instead, it was only fine-tuned using cross entropy loss and annotated images while the encoder was fixed. This operation was only performed before evaluating the performance. It should be noted that the projector was not used during evaluation as it only served as a means of training the encoder. The classifier was frozen at the time of fine-tuning the classifier. This ensured that the trained $\boldsymbol{v}$ vectors were unaffected. In all experiments, the classifier contained only a FC layer with an input size of 512 and an output size of K, and a softmax function.

Figure 3.3 shows the structure and relation of three modules during encoder training and classifier fine-tuning, where B is the size of pseudo multi-view batch, as introduced in Section 3.2.3.

3.2.3 Supervised Contrastive Loss Function

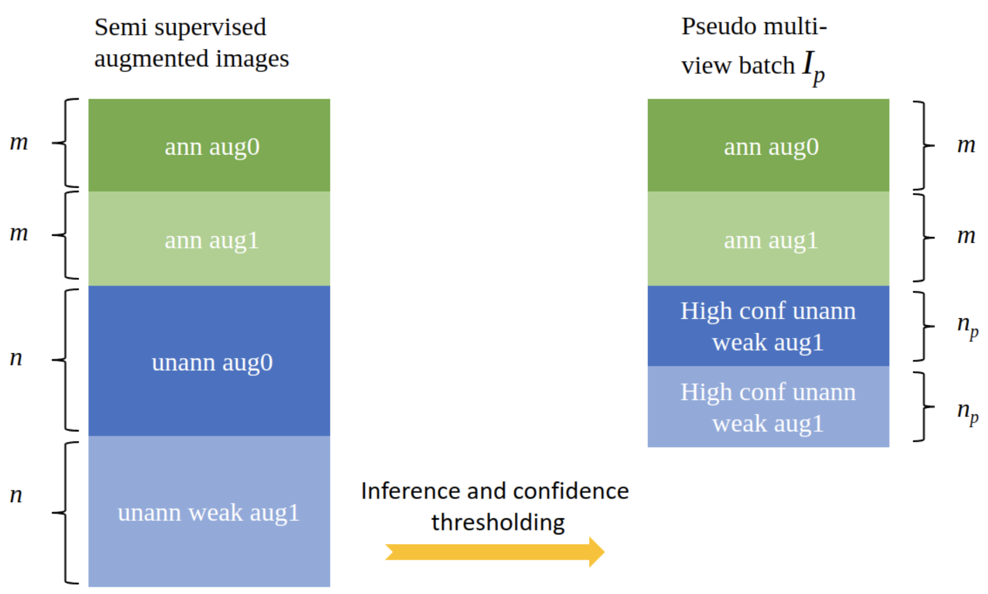
In fully supervised setting as Khosla et al. (2020), there were only annotated images. Two independent augmentations were applied to each annotated image, $\boldsymbol{x} = concate(\boldsymbol{x}_{aug0}, \boldsymbol{x}_{aug1})$, where $\boldsymbol{x}$ represented all the augmented images and $concate()$ was the concatenate operation. For a batch of m images, $\boldsymbol{x}$ had a size of $2m$ and was referred to as the multi-view batch.
$i \in I \equiv \{1 \ldots 2m\}$ was the indices of the multi-view batch. Forwarding the multi-view batch through the encoder and projector gave $\boldsymbol{v} = norm(Enc(\boldsymbol{x}))$, $\boldsymbol{z} = norm(Pro(\boldsymbol{v}))$. For each index i, all other indices composed the anchor set $A(i) \equiv I \backslash i$. The indices in $A(i)$ that originated from same category as i (which meant $\boldsymbol{y} = \boldsymbol{y_i}$) formed the positive pair indices set $P(i)$. It was straightforward that the index of augmented image which originated from same source image as $\boldsymbol{x_i}$ must be in $P(i)$. Following Khosla et al. (2020), the fully supervised contrastive loss took the form:

$$L^{\text{sup}} = \sum_{i \in I} L_i^{\text{sup}} = \sum_{i \in I} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)} \tag{3.2}$$

Herein, $|P(i)|$ is the cardinality of $P(i)$ and τ is a temperature parameter. It should be noted that the summation of positive pairs $\sum_{p \in P(i)}$ could also be within the log function. However, Khosla et al. (2020) proved that this causes the performance to deteriorate and was not adopted in this study. For each z_i, the supervised contrastive loss attracted its positive pairs and expelled its negative pairs.

3.2.4 Semi-Supervised Contrastive Loss function

The proposed method SemiCon introduced the supervised contrastive learning into semi-supervised setting by adding the high confidence unlabeled images to the training data and their predictions were treated as pseudo labels. At every training step, there were a batch of m annotated images $\boldsymbol{x}^{ann}$ and a batch of n unannotated images $\boldsymbol{x}^{unann}$. Two types of random augmentations, weak-augmentation and strong-augmentation, were performed on the images. The annotated images were strong-augmented twice $\boldsymbol{x}^{ann}_{aug0}$ and $\boldsymbol{x}^{ann}_{aug1}$ while the unannotated images were strong-augmented once $\boldsymbol{x}^{unann}_{aug0}$ and weak-augmented once $\boldsymbol{x}^{unann}_{waug1}$. Therefore, there were a total of $2m + 2n$ augmented images in the multi-view batch $\boldsymbol{x} = concate(\boldsymbol{x}^{ann}_{aug0}, \boldsymbol{x}^{ann}_{aug1}, \boldsymbol{x}^{unann}_{aug0}, \boldsymbol{x}^{unann}_{waug1})$. $i \in I \equiv \{1 \ldots 2m + 2n\}$ were the indices of $\boldsymbol{x}$. $\boldsymbol{z} = concate(\boldsymbol{z}^{ann}_{aug0}, \boldsymbol{z}^{ann}_{aug1}, \boldsymbol{z}^{unann}_{aug0}, \boldsymbol{z}^{unann}_{waug1})$ were the feature vectors of $\boldsymbol{x}$. The acquisition of $\boldsymbol{z}$ is introduced in Section 3.2.6. The probability distribution of the unannotated images was inferred using the feature vectors of their weak augmentation $\boldsymbol{z}^{unann}_{waug1}$. The inference function $infer()$ is introduced in Section 3.2.5. The unannotated images whose inferred probabilities were higher than a confidence threshold were merged with the annotated images to construct the pseudo multi-view batch, and their predictions were treated as their labels. The indices of the pseudo multi-view batch were I_p. If n_p unannotated images were pseudo labeled, then I_p had a size of $2m + 2n_p$. The pseudo multi-view batch was trained by following the same strategy as supervised contrastive learning. The construction of the pseudo multi-view batch is concluded in Figure 3.4.



Semi supervised augmented images
Pseudo multi-view batch I_p
m
ann aug0
m
ann aug1
n
unann aug0
n
unann weak aug1
Inference and confidence thresholding
ann aug0
m
ann aug1
m
High conf unann weak aug1
n_p
High conf unann weak aug1
n_p

Figure 3.4. Pseudo multi-view batch

The loss function is concluded as:

$$\begin{aligned}
&\boldsymbol{y}_{prob}, \boldsymbol{y}_{pred} = infer(\boldsymbol{z}_{waug1}^{unann}) \\
&\boldsymbol{y}[2m+1:] = Concatenate((\boldsymbol{y}_{prob}, \boldsymbol{y}_{prob})) \\
&I_{idx} = \{i+2m | y_{prob}[i] > conf\_thresh\} \cup \{i+2m+n | y_{prob}[i] > conf\_thresh\} \\
&I_p = \{1, 2 \dots 2m\} \cup I_{idx} \\
&L^{\text{sup}} = \sum_{i \in I_p} L_i^{\text{sup}} = \sum_{i \in I_p} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p / \tau)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a / \tau)}
\end{aligned} \tag{3.3}$$

where $\boldsymbol{y}_{prob}$ and $\boldsymbol{y}_{pred}$ are the probability and top-1 prediction of $\boldsymbol{z}_{waug1}^{unann}$, with a size of n, and $\boldsymbol{y}$ is the label of the multi-view batch $\boldsymbol{x}$.

3.2.5 Vector Based Inference

Contrastive learning optimizes the distance between feature vectors instead of the distance between each feature vector and its corresponding class center. The model only outputs the feature vector instead of predicted distribution. Meanwhile, the pseudo labeling process requires the prediction of unannotated images to select high confidence samples. Routinely, a classifier is trained to project the feature vector to the predicted distribution. However, in the proposed semi-supervised setting, the pseudo labeling process was at every training step and fine-tuning the classifier before every training step was not affordable. Therefore, it was necessary to find a method that inferences a feature vector through its relation with other feature vectors instead of its distance to the class center. To set up the foundation of the inference, a group of feature vectors $\boldsymbol{z}$ with the corresponding labels $\boldsymbol{y}$ known were stored in memory as a feature bank. The probability of new feature vectors could be inferred using the feature bank. The details of the feature bank are presented in Section 3.2.6.

A naive solution was to use the k-nearest-neighbor (KNN) algorithm. To predict the probability distribution of a feature vector, KNN finds the its nearest neighbors from the feature bank, and then make inference through the labels and distances of the nearest neighbors. However, to obtain more accurate predictions, a graph based label propagation algorithm originating from Zhou, Bousquet, Lal, Weston, and Schölkopf (2003) was chosen.

The vectors in a memory bank of size L_s were referred to as the support set $S := \{\boldsymbol{z_i}\}_i^{L_s}$, and the L_q vectors to be inferred were referred to as the query set $Q := \{\boldsymbol{z}_{L_s+i}\}_i^{L_q}$. The support vectors and query vectors were merged to construct a total set $Z = \{\boldsymbol{z_1}, \ldots, \boldsymbol{z_T}\}$ where $T := L_s + L_q$. A KNN graph of all vectors in Z was first built, as represented by a $T \times T$ matrix A:

$$A_{ij} := \begin{cases} [\boldsymbol{z_i}^T \boldsymbol{z_j}]_+^\gamma & \text{if } i \neq j \text{ and } \boldsymbol{z_i} \in KNN(\boldsymbol{z_j}) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where i and j are the row index and column index of A, $i \in [T]$ and $j \in [L_q]$. $KNN(\boldsymbol{z_j})$ returned the k nearest neighbors of $\boldsymbol{z_j}$ within set Z. Inner product was treated as the similarity metric here. Since $\boldsymbol{z}$ vectors were normalized, the inner product was equivalent to cosine similarity. $\gamma > 1$ was a hyper-parameter. The A matrix was sparse since k was much smaller than T. The A matrix was symmetrized and normalized as

$$\begin{aligned} W &:= 0.5 \times (A + A^T) \\ D &= diag(W 1_T) \\ W &:= D^{-1/2}(W)D^{-1/2} \end{aligned} \tag{3.5}$$

where 1_T is the transpose of a vector with full ones, and D is a diagonal matrix where its diagonal elements are the row sum of W. The labels of support set and query were merged as a label matrix Y of shape $T \times K$:

$$Y_{ij} := \begin{cases} 1 & \text{if } i \neq L_s \text{ and } y_i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

Recall K is the number of classes defined in Section 3.2.1. The first L_s rows of Y were one-hot labels of the support set S while the remained L_q rows were the probabilities of the query set Q. Since the labels of query set were unknown, the remaining L_q rows were all zero. P^* was a probability matrix of shape $T \times K$ where its values $P^*[t,k]$ were the probability of sample t belonging to category k. α in $(0,1)$ was a parameter that measures the impact between neighborhood samples. The P^* was updated by iterating over the graph as:

$$\begin{aligned} P^*(0) &= Y \\ P^*(iter+1) &= \alpha W P^*(iter) + (1-\alpha)Y \end{aligned} \tag{3.7}$$

until P^* converged to $P = \lim_{iter \to \infty} P*(iter)$, where $iter$ is the iteration number.

Hence

$$P^*(iter) = (\alpha W)^{iter-1} Y + (1-\alpha) \sum_{i=0}^{iter-1} (\alpha W)^i Y \tag{3.8}$$

The eigenvalue of W was in $[-1,1]$ and α in $(0,1)$, and therefore

$$\begin{aligned} &\lim_{iter \to \infty} (\alpha W)^{iter-1} = 0 \\ &\lim_{iter \to \infty} \sum_{i=0}^{iter-1} (\alpha W)^i = (I - \alpha W)^{-1} \end{aligned} \tag{3.9}$$

$$P = \lim_{iter \to \infty} P^*(iter) = (1-\alpha)(I-\alpha W)^{-1}Y \tag{3.10}$$

It should be noted that $1-\alpha$ was a constant in $(0,1)$ and did no affect the classification result of the P matrix. Therefore, the calculation of P could be simplified as:

$$P = (I-\alpha W)^{-1}Y \tag{3.11}$$

The prediction of the query set Q could be made by the last L_q rows of P. To convert P as the probability matrix, the negative values in P were set to zero, and each row was normalized so that the row sum equals one. Then, each element in P represented the predicted probability of the corresponding sample in the corresponding category.

This algorithm originated from the spreading activation network from experimental psychology. It can be described as a process of diffusion along the graph, where the category information flow through the graph to all vertices starting from the support set and reaches an equilibrium. This graph based inference process is illustrated in Figure 3.5 where circles are the support set and rectangles are the query set.

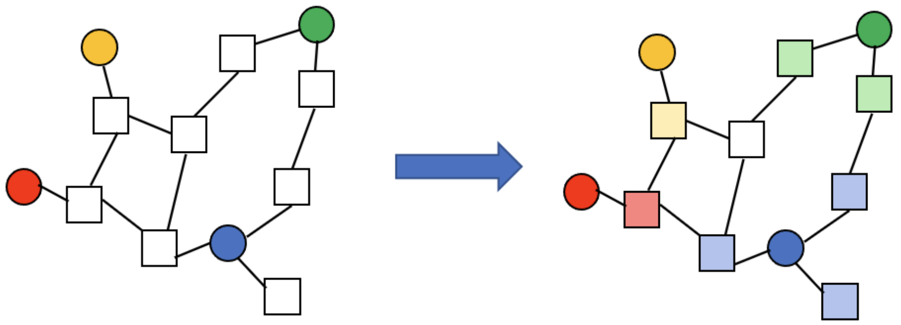


Figure 3.5. Graph based inference

In the proposed SemiCon, the query set Q was comprised of the $\boldsymbol{z}$ vectors of the augmented unannotated images, so that $L_q = 2n$. The support set was the feature bank instead of $\boldsymbol{z}$ vectors of the augmented annotated images, and hence $L_s \neq 2m$.

The inference function on feature vectors of all weakly augmented unannotated images $\boldsymbol{z}_{waug1}^{unann}$ could be expressed as:

$$\begin{aligned} P &= calculate\_P(S = feature\_bank, Q = \boldsymbol{z}_{waug1}^{unann}) \\ pred, prob &= infer(\boldsymbol{z}_{waug1}^{unann}) \\ &= argmax(P[L_s :, :], dim = 1), max(P[L_s :, :], dim = 1) \end{aligned} \tag{3.12}$$

where $calculate\_P$ is the summary from Equation 3.4 to Equation 3.11. It should be noted that the inference of each weakly augmented unannotated image could not be separated, since the inference were proceeded on a graph where samples were connected. This would reduce the accuracy of inference.

It should also be noted that the pseudo labeling process could be further optimized if the distribution of the unannotated images was known or could be estimated. In this case, the P matrix could be optimized using the *Sinkhorn-Knopp* algorithm from Knight (2008).

3.2.6 Feature Bank and Momentum Encoder

The support set S was stored as a feature bank. It is natural that a larger feature bank achieves a better performance. Therefore, using only the annotated images from each training step was insufficient. The feature vectors from multiple training steps were used to fill in the feature bank. However, the feature vectors from different training steps were inferred from different model parameters, which caused a mismatch within the feature bank as well as between the feature bank and current training images. Therefore, the vector space had to be involved gradually to maintain the consistency of feature vector space between different training steps.

Inspired by He, Fan, Wu, Xie, and Girshick (2020), a momentum encoder $MEnc()$ which had the same architecture as the pipeline of encoder and projector $Pro(norm(Enc()))$ was maintained. The parameters of momentum encoder were initiated with the same parameters of encoder and projector, and then it was updated as:

$$\theta_m \leftarrow moment\,\theta_m + (1 - moment)\theta_{ep} \tag{3.13}$$

where θ_m is momentum encoder parameter, θ_{ep} is the parameter of encoder and projector at the current training step, and $moment \in [0,1)$ is a momentum coefficient. The momentum encoder was not updated through back-propagation and there was no gradient flow during forwarding operation. $m = 0.999$ in all experiments. The $\boldsymbol{x}^{ann}_{aug0}$ and $\boldsymbol{x}^{unann}_{aug0}$ were fed into the encoder and projector, while the $\boldsymbol{x}^{ann}_{aug1}$ and $\boldsymbol{x}^{unann}_{waug1}$ were fed into the momentum encoder. The acquisition of $\boldsymbol{z}$ can be presented as:

$$\begin{aligned}
&\boldsymbol{z}^{ann}_{aug0} = Pro(norm(Enc(\boldsymbol{x}^{ann}_{aug1})))\\
&\boldsymbol{z}^{unann}_{aug0} = Pro(norm(Enc(\boldsymbol{x}^{unann}_{waug1})))\\
&\boldsymbol{z}^{ann}_{aug1} = MEnc(\boldsymbol{x}^{ann}_{aug1})\\
&\boldsymbol{z}^{unann}_{waug1} = MEnc(\boldsymbol{x}^{unann}_{waug1})\\
&\boldsymbol{z} = concate(\boldsymbol{z}^{ann}_{aug0}, \boldsymbol{z}^{ann}_{aug1}, \boldsymbol{z}^{unann}_{aug0}, \boldsymbol{z}^{unann}_{waug1})
\end{aligned} \tag{3.14}$$

Initially, the feature bank was empty. It was filled with $\boldsymbol{z}^{ann}_{aug1}$ and their corresponding labels at every training step. No inference on unannotated images was performed, and the pseudo multi-view batch contained only the augmentations of annotated images. The model (projector and encoder) was trained by only annotated images and the momentum encoder was updated by the model. This continued until the feature bank was full. Then, $\boldsymbol{z}^{unann}_{waug1}$ was inferred by the feature bank, and high confidence unannotated images were selected. $\boldsymbol{z}^{ann}_{aug1}$, the selected high confidence $\boldsymbol{z}^{unann}_{waug1}$, and their corresponding labels were fed into the feature bank. Meanwhile, the $\boldsymbol{z}^{unann}_{aug0}$ and $\boldsymbol{z}^{unann}_{waug1}$ of the selected images were merged with $\boldsymbol{z}^{ann}_{aug0}$ and $\boldsymbol{z}^{ann}_{aug1}$ to form the pseudo multi-view batch. The model (projector and encoder) was then trained by the pseudo multi-view batch and the momentum encoder was updated. The feature bank was kept at a fixed size and the oldest features and their labels popped out in each update.

The training of Semicon is concluded as Algorithm 3.1. Figure 3.6 presents the training process and gradient flow when the feature bank is full.

Algorithm 3.1 SemiCon

```
1: for $i$ in $total\_iters$ do
2:     Initial or update momentum encoder $MEnc$
3:     Iterate annotated dataloader and unannotated dataloader to get $\boldsymbol{x}^{ann}$ and $\boldsymbol{x}^{unann}$
4:     Perform augmentation to get $\boldsymbol{x} = concate(\boldsymbol{x}^{ann}_{aug0}, \boldsymbol{x}^{ann}_{aug1}, \boldsymbol{x}^{unann}_{aug0}, \boldsymbol{x}^{unann}_{waug1})$
5:     Forward to get $\boldsymbol{z} = concate(\boldsymbol{z}^{ann}_{aug0}, \boldsymbol{z}^{ann}_{aug1}, \boldsymbol{z}^{unann}_{aug0}, \boldsymbol{z}^{unann}_{waug1})$    ▷ Equation 3.14
6:     if feature bank is full then
7:         Inference $\boldsymbol{z}^{unann}_{waug1}$ using feature bank    ▷ Equation 3.12
8:         Obtain pseudo multi-view batch index $I_p$ by apply threshold on inference probability
9:         fill feature bank with $\boldsymbol{z}^{ann}_{aug1}$, pseudo labeled $\boldsymbol{z}^{unann}_{waug1}$ and their label
10:    else
11:        pseudo multi-view batch index $I_p = 1 \ldots 2m$
12:        fill feature bank with $\boldsymbol{z}^{ann}_{aug1}$
13:    end if
14:    Train the projector $Pro$ and encoder $Enc$ by loss function
15: end for
```

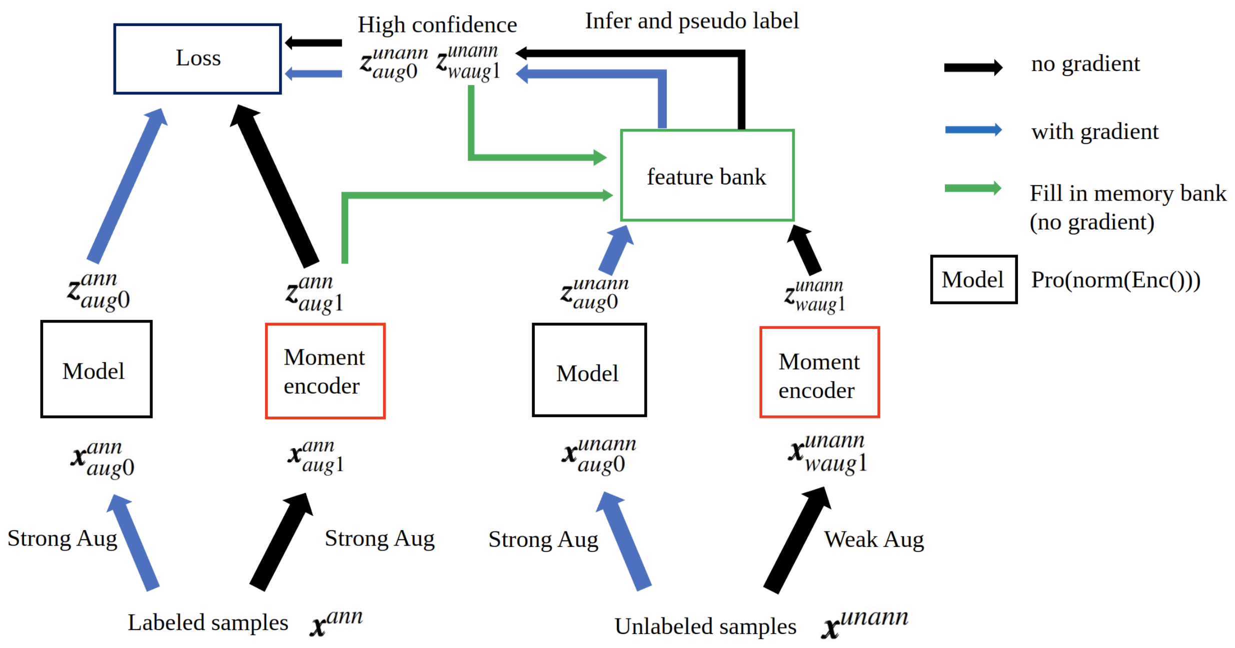


Figure 3.6. Training process and gradient flow

CHAPTER 4. ANOMALY DETECTION OF INDUSTRIAL IMAGES

This chapter presents the detailed algorithm design and experiments of the anomaly detection study (Tang et al., 2021). It first introduces the dataset and the data processing strategy. Then, the proposed algorithm is presented comprehensively and the design of each process is presented as well. The proposed approach is compared with several other approaches. The generalizabilty of the proposed approach and how to adapt it to other inspection scenarios are discussed as well.

4.1 Dataset and Hardware

The proposed algorithm was tested on a dataset from a real-world aluminum alloy die-casting core inspection scenario. It was provided by Fiat-Chrysler Kokomo Casting Plant in Kokomo, Indiana, USA. Die casting is a process of injecting high-temperature liquid metal, normally aluminum alloys or magnesium alloys, into a shaped mold. The liquid metal solidifies into the desired geometry after cooling down. Cores are the pieces of steel used to create negative spaces in the geometry. These cores are often broken due to the high pressure during the injection process and cause failure of the product. The products are scanned by the X-Ray machine and visually inspected by humans.

The dataset contained a total of 236 images. There were 122 good images and 114 defective images. All of the images had a resolution of 2048×2048. Figure 4.1 presents an example of a good image. The center black circle area annotated by a red box was the area of interest for the inspection. There were three cores, including the center core, the slot core, and the small core, in the area of interests. These cores were prone to disappearance and shortage. Figure 4.1 presents the location of each core, and four types of defects. It should be noted that different types of defect could coexist, thus causing additional types of defects. However, from the perspective of inspection, there were no differences between these types of defect since any defective product would be scrapped and remelted.

To ensure the rigorousness of the research, the dataset was randomly split into training set, validation set, and test set for three trials in all the experimental settings. As introduced in Section

3, only good images existed in training set and validation set, while the test set had a mix of good images and defective images. The proposed method was tested on the training set of four different sizes: 10 images, 20 images, 30 images, and 40 images. A validation set comprised of 10 images was used in all experiments. The remaining images were assigned to the testing set. The detailed composition of the three sets in each experiment is presented in Table 4.1. The test set held an approximated equilibrium between the good images and defective images, which reduced the bias of evaluation.

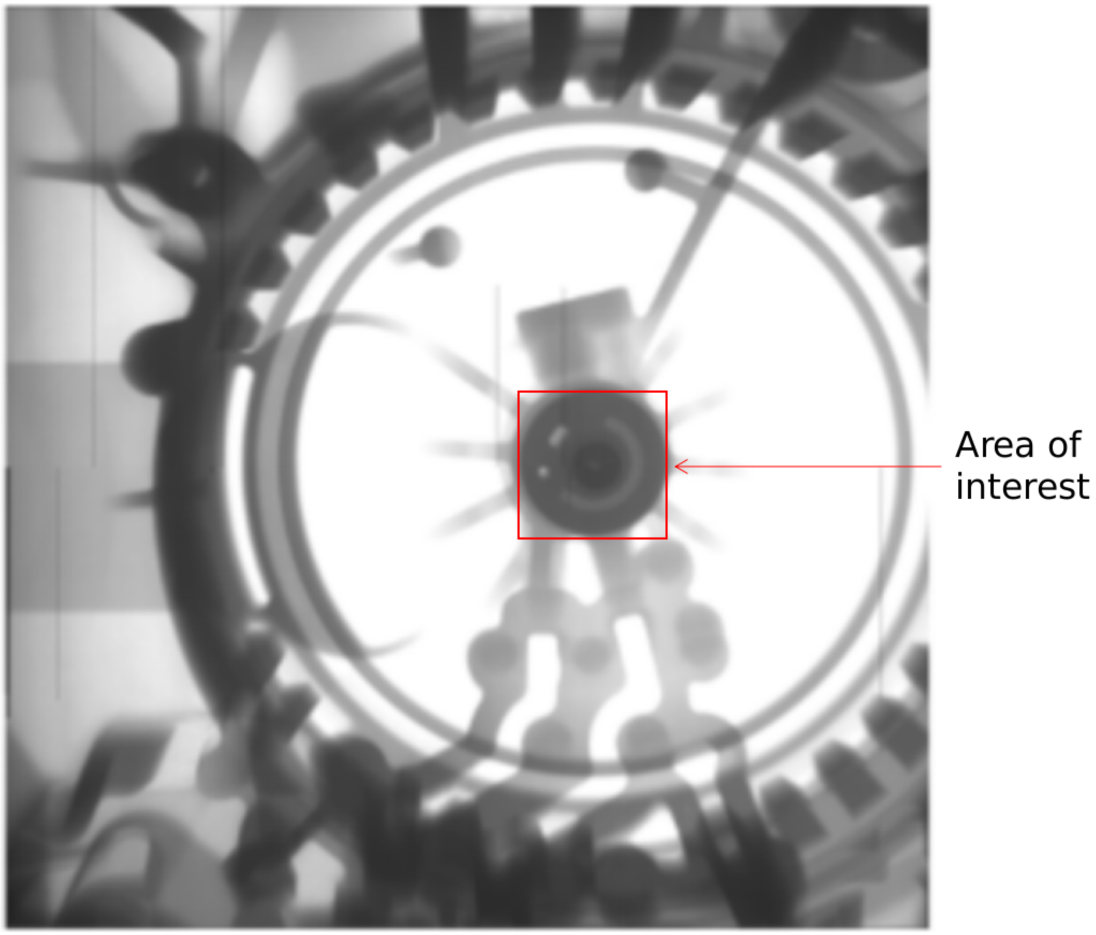


Figure 4.1. An example of the X-ray scanning of the product (Tang et al., 2021)

All of the experiments were performed on a workstation with an i7-9700k CPU, an Nvidia 2080s GPU with 8 GB of memory, and a total of 32 GB DDR4 RAM. Pytorch 1.3, opencv-python 4.4, Python 3.6, and Scikit-Learn 0.21.3 were used.

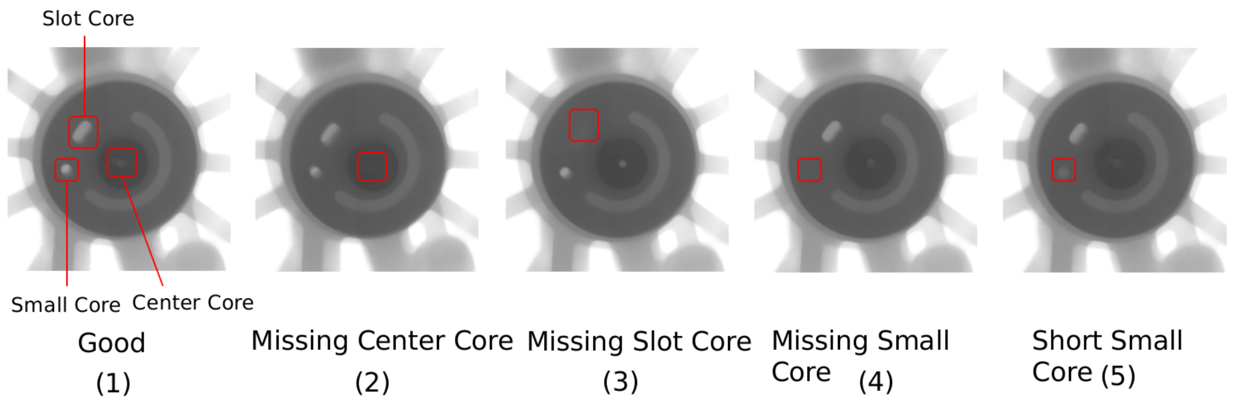


Figure 4.2. Example of a good image and different categories of defects (Tang et al., 2021)

Table 4.1. *Dataset setting (Tang et al., 2021)*

Setting	1	2	3	4
Training ”Good”	10	20	30	40
Validation ”Good”	10	10	10	10
Testing ”Good”	102	92	82	73
Testing ”Defective”	114	114	114	114
Testing ”Total”	216	206	196	186
Total	236	236	236	236

4.2 Automatic Preprocessing

The equipment setting in the foundry caused several white dots and black lines at the same location of each image. These noises were removed by interpolation using surrounding pixels. The castings were transferred into the X-Ray scanning machine through operator controlled robot arm, thus causing orientation and position inconsistency between each image. An automatic preprocessing algorithm was first designed to locate the area of interest.

The area of interest had a circle shape design, so that it could be identified by the circle Hough transformation (CHT). The area of interest in each image had a size of approximately 355×355 pixels. This size was increased to 465×465 pixels to compensate for rotation augmentations. Afterwards, the area of interest was cropped as a squared image and treated as the input of the proposed algorithm. The detailed pipeline is presented in Figure 4.3.

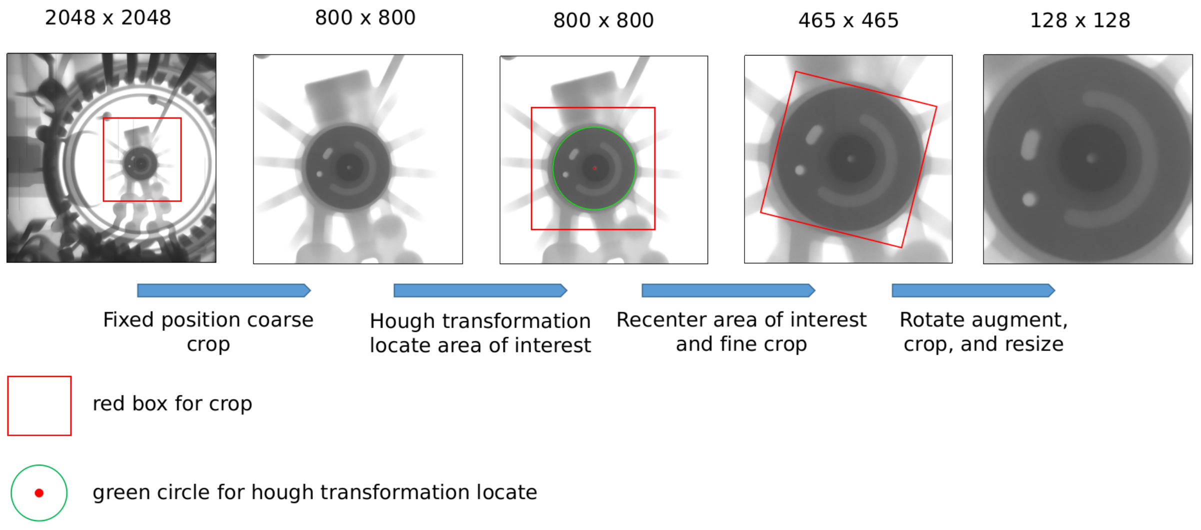

2048 x 2048
800 x 800
800 x 800
465 x 465
128 x 128
Fixed position coarse crop
Hough transformation locate area of interest
Recenter area of interest and fine crop
Rotate augment, crop, and resize
red box for crop
green circle for hough transformation locate

Figure 4.3. The automatic preprocessing pipeline (Tang et al., 2021)

4.3 Network Architecture

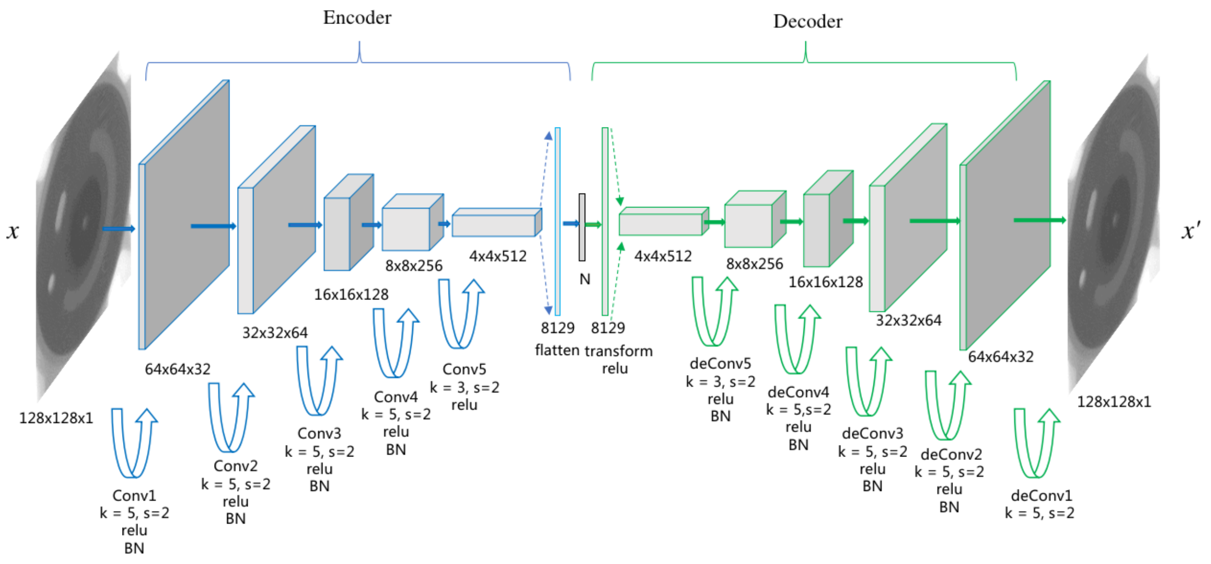


Figure 4.4. Network structure of the CAE model (Tang et al., 2021)

Figure 4.4 presents the detailed architecture of the CAE. There were five convolutional layers in the encoder and five deconvolutional layers in the decoder, thus resulting in a symmetric architecture. The bottleneck was constructed by two fully connected layers, with one used to downscale the feature vector and the other used to upscale the feature vector. Reshapes were proceeded before and after the bottleneck to enable transfer between convolutional/deconvolutional layers and fully connected layers. All convolutional and deconvolutional layers had a stride of two and zero padding. Batch normalization and relu activation functions were performed as well. To study the impact of the bottleneck size N, it was set to 5, 10, 20, and 100 respectively. The network took the images of $128 \times 128 \times 1$ as input and outputted the images with the same dimension.

4.4 Training

The automatically preprocessed images were 465×465 in resolution. The images randomly rotated within ten degrees as augmentation to simulate different placements of the product in the X-Ray. The center area, which was the area of interest, was then cropped into 325×325, and resized into 128×128. All the pixel values were normalized with a mean of 0.5 and a standard deviation of 0.5. The images were inputted into the network after these processes.

The Adam optimizer with a learning rate of 0.001 and a batch size of two was used to train the network for 3000 epochs. For a training set size of 10, 20, 30, and 40 images, the training took approximately 5, 10, 15, and 20 minutes.

4.4.1 Impact of bottleneck size

Setting up the model architecture required a proper bottleneck size to be selected at first. The impact of the bottleneck size was studied by training the model with a bottleneck size N of 5, 10, 20, and 100, respectively. There were 30 images in the training set and 10 images in the validation set in these experiments. Figure 4.5 presents the training loss and validation loss averaged over every 200 epochs of different bottleneck sizes. It should be noted that the Y-axis is on log scale, which significantly increases the distance between small values.

In general, a smaller bottleneck gave a higher constrain for the CAE and resulted in a more significant loss, which led to low quality output. However, if the bottleneck size was too large, the model would be likely to over-fit on the data and copy the input into the output. This would cause the defects to be copied into the output and they could not be inspected as a result. In the experiments, a bottleneck size of 5 gave a slightly higher loss curve compared with other bottleneck sizes, while there were no significant differences between the loss curves of bottleneck sizes of 10, 20, and 100. During the first 1000 epochs, the loss cures dropped rapidly from 0.3000 to 0.0001. Since then, the loss decreased at a lower rate. After 2000 epochs, the loss dropped insignificantly and started oscillating. The validation loss was not reversed during training, which suggested no overfitting.

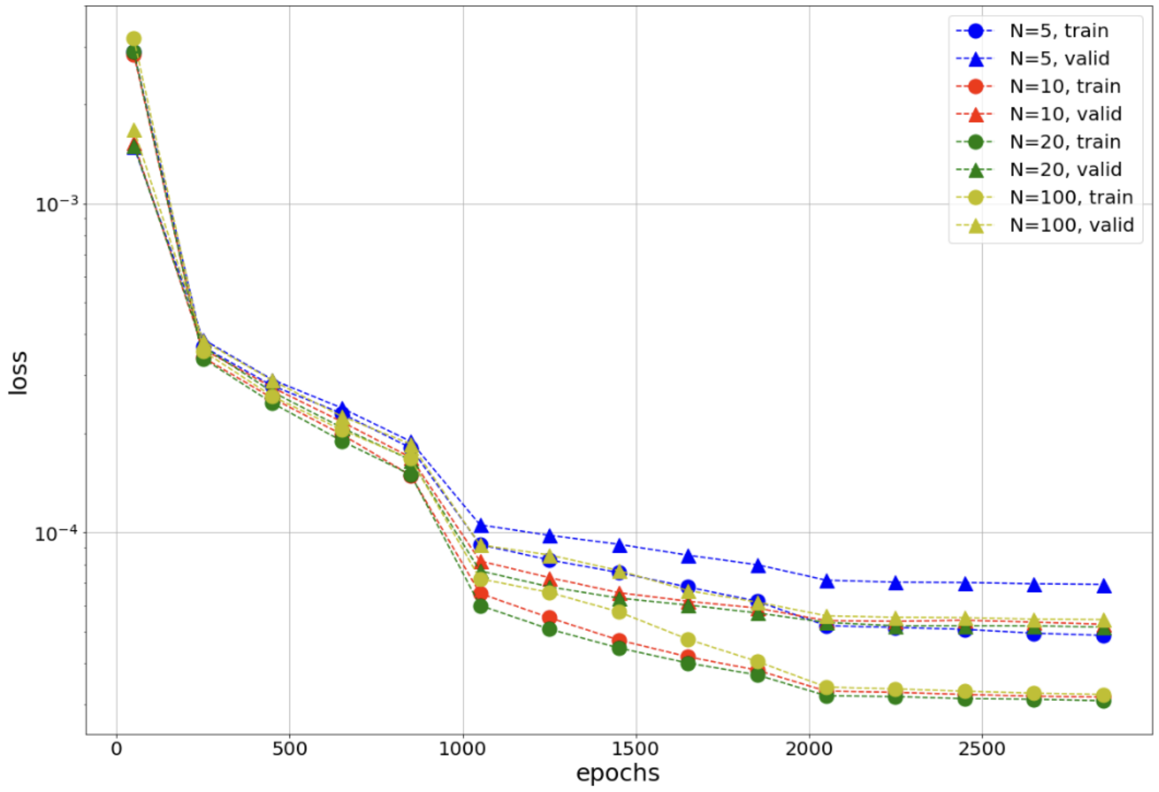


Figure 4.5. Loss versus epochs for different bottleneck size N (Tang et al., 2021)

Since no overfitting was observed, bottleneck N was set to 100. It should be noted that the impact of variation in training set size in Table 4.1 was also tested. However, the corresponding loss curves of these experiments are not presented since there are hardly any differences in trend from the presented loss curve. The average loss of last 100 epochs for all training set sizes with $N = 100$ is presented in Table 4.2. A larger training set provides more information to the model, thus reducing both the training loss and the validation loss.

Table 4.2. *The mean loss value of last* 100 *epochs when* $N = 100$ *(Tang et al., 2021)*

Training Size	Train Loss	Validation Loss
10	7.1×10^{-5}	3.2×10^{-4}
20	5.0×10^{-5}	2.1×10^{-4}
30	3.2×10^{-5}	5.2×10^{-5}
40	2.7×10^{-5}	4.3×10^{-5}

In conventional computer vision datasets like CIFAR, ImageNet, and COCO, there is a high diversity among the images in the same categories, which makes overfitting an essential challenge. However, the die casting products without defect have a high level of consistency, which results in low divergence between the trend of the training and validation curves. This suggests that the training set represents the whole population well. Such behaviour is common in many other manufactured products. Modern manufacturing processes are often standardized to create almost identical products in case of no failure. The low variation among the manufactured products satisfies the designed tolerance. This makes it possible to train a proper model with a limited amount of data since the high consistency among the good products ensures the high representability of the training set. It should be noted that such great feature does not hold using a conventional classification, since a classifier learns from both good and defective products. Besides, there can be a significant diversity among different defects. Bian et al. (2016) serves as a great example of how diverse defective products can be. They proposed a complex multi-scale fusion approach to suppressing overfitting.

4.4.2 CAE Output and the difference images

The images from the test set were inputted into the trained CAE model. The difference between input image and output image was utilized to measure the divergence between input images and good images. If the divergence was small, the input image was inspected as good; otherwise, it was inspected as defective. The model was only trained with good images, which led to a lack of the ability to perform well on defective images. More specifically, a difference image was first calculated by a pixel-wise subtraction of the original image with the output image. Then, it was converted into absolute value. Figure 4.6 presents an example of the input image, output image, and difference image of good images and several types of defective images. Apparently, the difference image shows trivial values in any area of the good images. However, for the defective images, the difference image shows trivial values in the good area and none-trivial values in the defective area. This enabled both classification and localization of the defect.

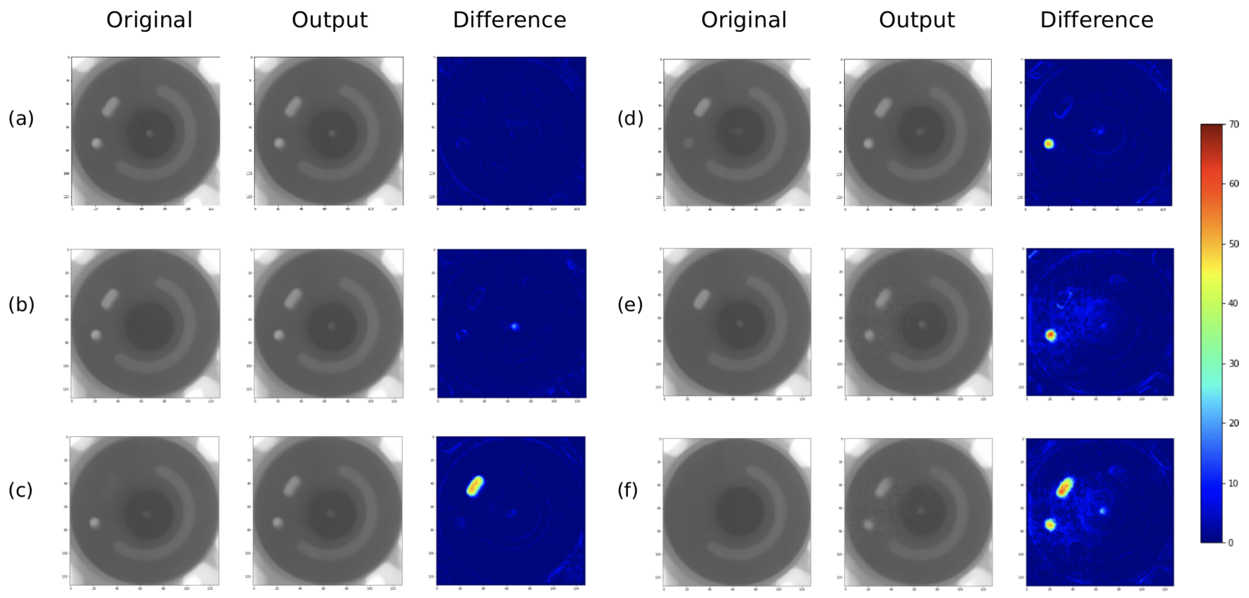


Figure 4.6. Original image, CAE output image, and difference images (Tang et al., 2021)

A basic strategy of distinguishing between good and defective images is to perform template matching. Template matching averages all the training images to construct a template image, which stands for the standard good image. One can inspect an image by comparing it with the template image. However, the template image constructed using such strategy was of poor quality in this scenario. The main reason for this was that products were transmitted into X-Ray machine with different orientations and placements, which caused blurriness on the template image. Figure 4.7 compares the template image and the output image of the model. The first row presents a good input image, the CAE output image, the difference image between input image and CAE output image, and the close view of rectangle area in the output image. The second row presents the template image, the difference image between input image and template image, and the close view of the rectangle area in the template image. Although the output image and template image were similar to each other, they were intrinsic in details. The averaging out over all training images apparently caused a significant blurriness at geometry boundaries and the "two ring" structure was completely eliminated in the template image. The difference image between the input image and output image had little value, while the difference image between the input image and template image had high values over the geometry boundaries. Using the template image instead of model output reduced the quality of the difference image significantly.

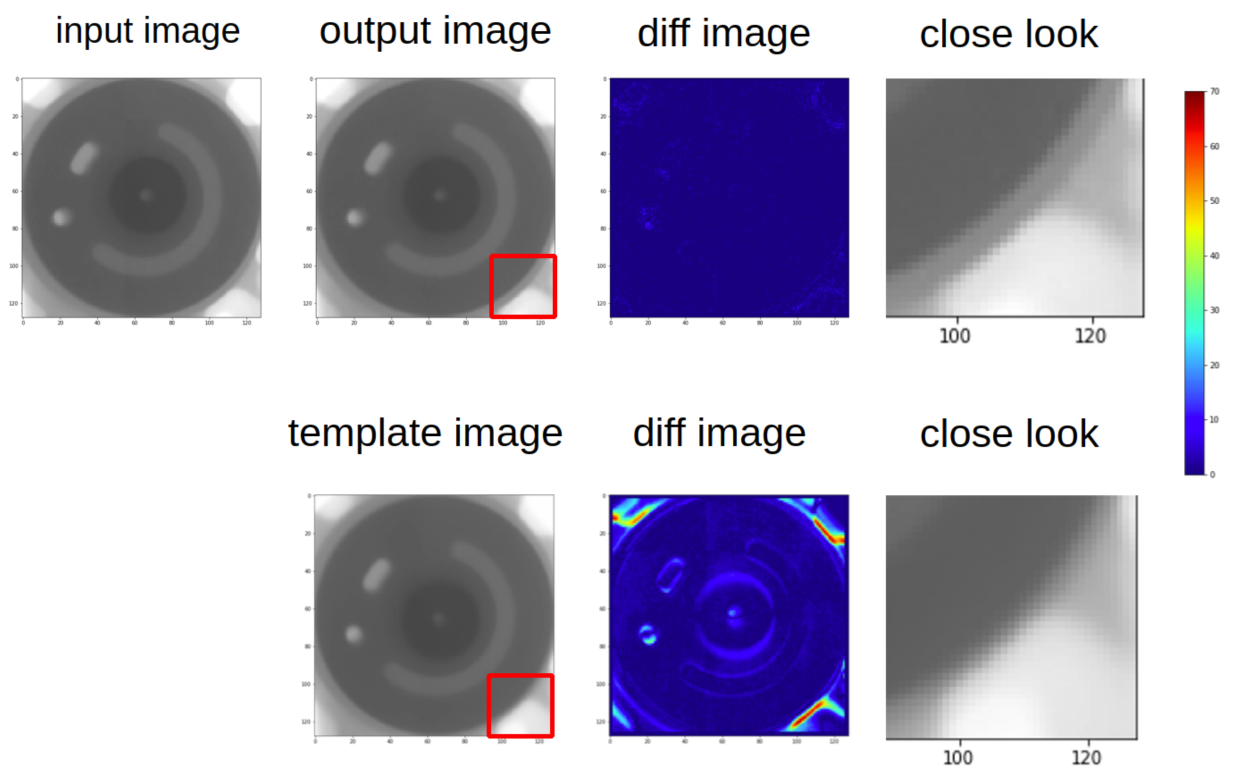


Figure 4.7. CAE output image versus template image.(Tang et al., 2021)

Despite the fact that the defect was obvious in the difference image, it was still complicated to recognize the defect automatically. The defect often composed only a small portion of the whole difference image, and therefore, it would be averaged out if only the mean value of the whole difference image was measured.

4.4.3 Noise analysis

Most model noises in the difference image were located at the geometry boundaries. It was observed that the missing center core images had a significantly smaller size and less significant brightness compared to other cores. This caused the defects of missing center core images to be blurring and had low values on the corresponding difference images. Therefore, these defects were likely to be mistaken for the noise at geometry boundaries. Figure 4.8 presents an example of such noise on a missing center core image. The CAE generated a blurry center core as it learned from good images, and showed the corresponding defect on the difference image. However, there were also obvious model noises at the geometric boundaries, and the strength of defect could not surpass these noise.

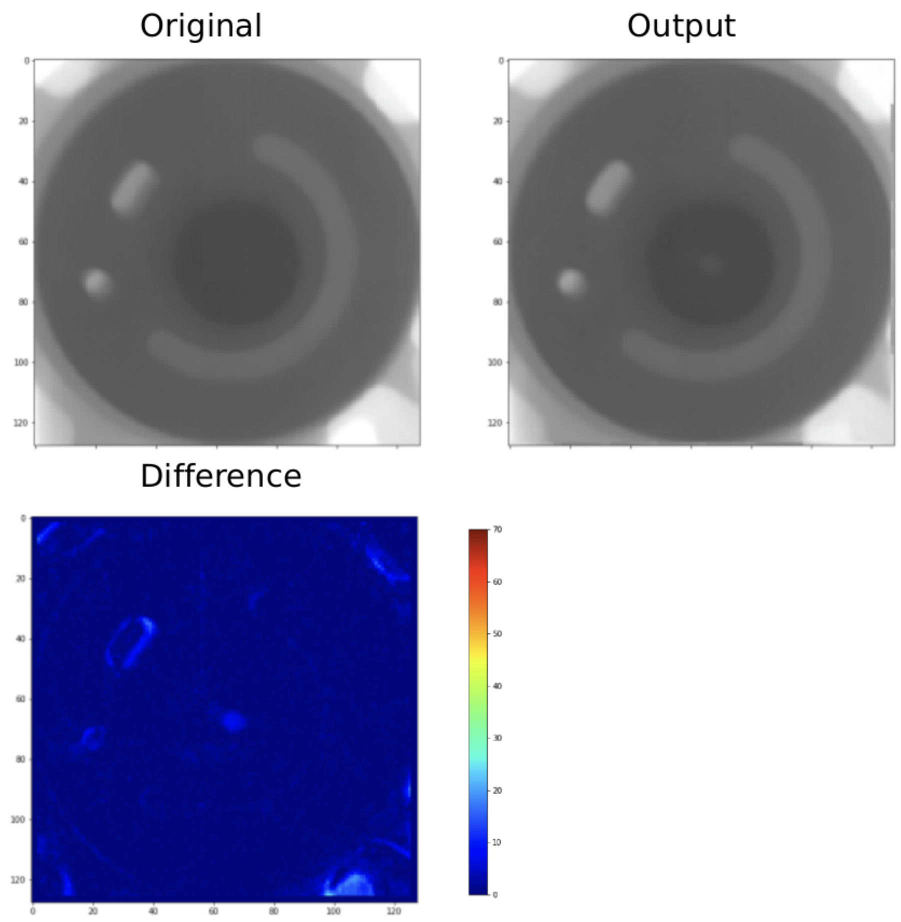


Figure 4.8. Example of model noise (Tang et al., 2021)

To make things worse, the center core could be ambiguous in some good images, which was acceptable according to the manufacturing standard. The CAE often failed to generate a perfect center core for these images, and caused model noise in the center core area of the corresponding difference images. This was universal for dimension reduction algorithms which tended to keep the most essential information while ignoring those minor features. Such model noise could be mistaken for the defects of a missing center core image. Notably, when human compares the original image with the output image, the missing center stands out at the first glance. However, the difference image presented that the difference at center core was less significant than the model noise at geometry boundaries. Human vision is sensitive to the non-uniformity in pixel values in a region but less so to gradual changes.

4.4.4 Inspection Process Parameters

According to Section 4.4.2 and Section 4.4.3, there are two main issues, the averaging effect and the model noise at geometric boundaries. To address these issues, a two-step inspection process was proposed in Section 3.1.3. This section introduces the selection of all parameters.

In the first step, a 3×3 sliding window with a stride of 3 was applied on the difference image. The small size of the sliding window was purposed to accommodate the small size of the cores. Such small cores would be passed by larger windows. The mean value of pixel values within the window was measured. Once the mean value exceeded a coarse threshold, the corresponding original image was recognized as defective and failed the inspection. Conversely, if the mean value was lower than the threshold, the corresponding original image succeeded the first step and would be passed to the second step. The coarse threshold was set as 31 to suppress most model noises at the geometric boundary. There was another fine threshold in step two, and the trade-off between these two thresholds was discussed in Section 4.4.6.

In the second step, the gradient operator was first applied to capture the gradient image. Instead of the universal 3×3 sober operator, a larger 7×7 operator from E. R. Davies (2012) was chosen. The main reason for this was that sober operator created narrow edges and only filters out a limited amount of model noises. The 7×7 operator is presented as follows:

$$\begin{bmatrix} 0.000 & 0.000 & -0.191 & 0.000 & 0.191 & 0.000 & 0.000 \\ 0.000 & -1.085 & -1.000 & 0.000 & 1.000 & 1.085 & 0.000 \\ -0.585 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 0.585 \\ -1.083 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 1.083 \\ -0.585 & -2.000 & -1.000 & 0.000 & 1.000 & 2.000 & 0.585 \\ 0.000 & -1.085 & -1.000 & 0.000 & 1.000 & 1.085 & 0.000 \\ 0.000 & 0.000 & -0.191 & 0.000 & 0.191 & 0.000 & 0.000 \end{bmatrix} \tag{4.1}$$

This operator had a circle-shaped weight distribution and a larger size than sobel operator, which resulted in a better gradient estimation (E. Davies, 1984). E. R. Davies (2012) introduced more details on the mathematical induction and application of these operators. Using this operator enlarged the edge areas and captured more model noises at geometric boundaries. To apply the operator at the image boundary, the images were padded with three additional pixels with value 255. Since all the core locations were not at the image boundary, no additional defects would be introduced. Figure 4.9 presents a missing core image and its gradient image. It can be noticed that the geometric boundaries are captured in the gradient image.

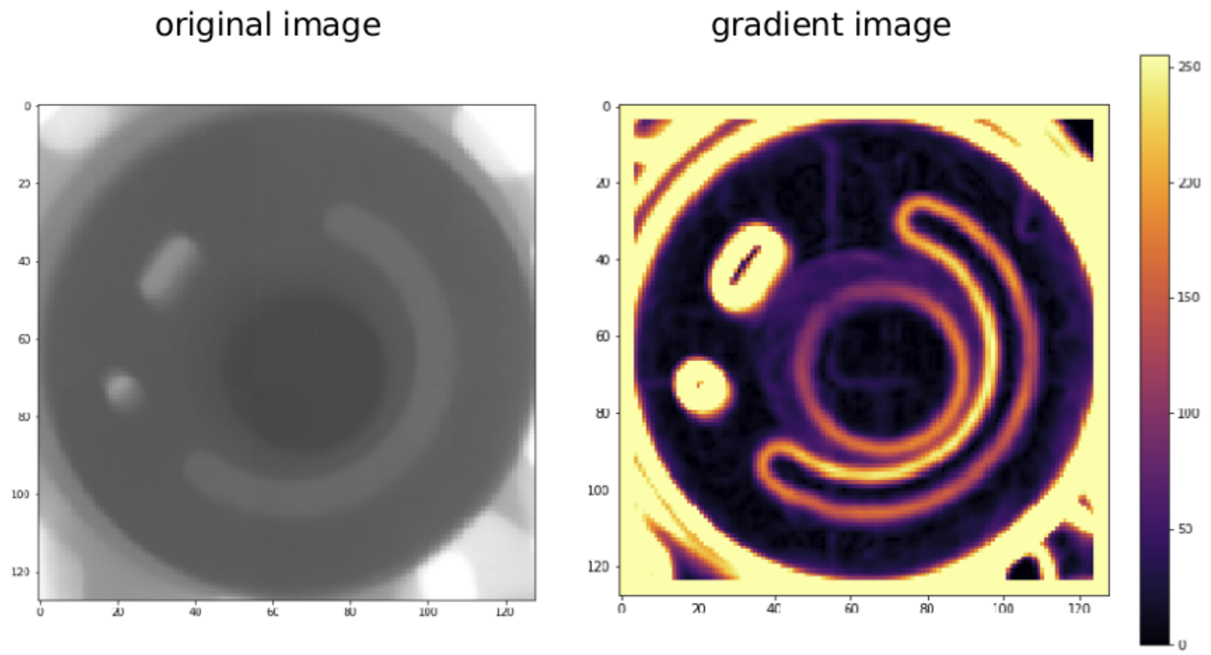


Figure 4.9. The original image and gradient image (Tang et al., 2021)

A 5×5 sliding window was used instead of a 3×3 sliding window in the first step. Since the major defects had already been filtered out in the first step, a larger window size was necessary to neglect the model noise in the window. To completely remove those noise-prone areas, the edge threshold was set to 70, which only preserved the darkest areas in the gradient image. As described in Section 3.1.3, both the mean pixel value of difference image and that of gradient image within the sliding window were checked. The original image was identified as defective only when the mean value of the gradient image was below the edge threshold and that of the difference image was above the fine threshold.

4.4.5 Evaluation Metrics

Evaluation metrics are introduced before the discussion about threshold trade-offs. Confusion matrix is a regular evaluation metric in machine learning. Table 4.3 presents the contents of a binary confusion matrix. True Negative (TN) refers to the number of defective images predicted as defective image, while False Positive (FP) refers to the number of defective images predicted as good image. Similarly, False Negative (FN) refers to the number of good images predicted as defective image, while True Positive (TP) refers to the number of good images predicted as good image. It should be noted that in this research, the good products were referred to as “positive” and defective products were referred to as “negative”, which was contradictory to the medical field where having a disease is referred to as “positive” and having no disease is referred to as “negative”.

Table 4.3. *Confusion matrix*

	Predicted Negative	Predicted Positive
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

Apparently, the goal was to maximize TP and TN while reducing FP and FN. Accuracy, precision, and recall were used to measure the performance of the proposed model. *Accuracy* is $(TN+TP)/(TN+TP+FN+FP)$, which measures the portion of correct prediction among all samples. *Precision* is $TP/(TP+FP)$, which measures the portion of correct predictions among all samples predicted as good. *Recall* is $TP/(TP+FN)$, which measures the portion of correct predictions among all the samples that are actually good. All the math notations are summarized in Table 4.4.

Table 4.4. *Evaluation metrics*

Metrics	Definition
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
Accuracy	$\frac{TP+TN}{TN+FP+FN+TP}$

4.4.6 Threshold and performance

Both the coarse threshold in step one and fine threshold in step two directly affected the model precision and recall. A higher threshold corresponds to a higher standard and results in a higher precision but lower recall. In contrast, a lower threshold corresponds to a lower standard and results in a higher recall but lower precision.

Precision was favored over recall in die casting inspection scenario. In other words, to avoid the amount of defective products for passing the inspection (FP), making sacrifice by having more good products to fail the inspection (FN) was preferred. The assumption here was that providing defective products to customers would result in greater economy losses and even safety concerns.

The thorough threshold deciding process followed by the first trial of experiment setting three, which included 30 images in the training set, was presented. The threshold deciding process of all the trials under all dataset settings followed the same idea.

In the first step of inspection, a coarse threshold was applied, and its impact on the FP and the FN was studied. Figure 4.10 presents the FP and FN with a coarse threshold varying from 25 to 39. It can be discovered that with an increase of the coarse threshold value, the number of FP rises while the number of FN drops. To obtain the minimum FP and reduce the FN, the coarse threshold was set to 31, which led to a FP of 22 and a FN of two.

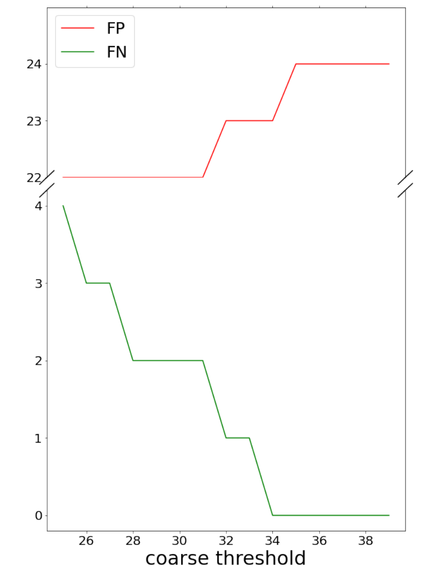


Figure 4.10. False positives and false negatives versus coarse threshold (Tang et al., 2021)

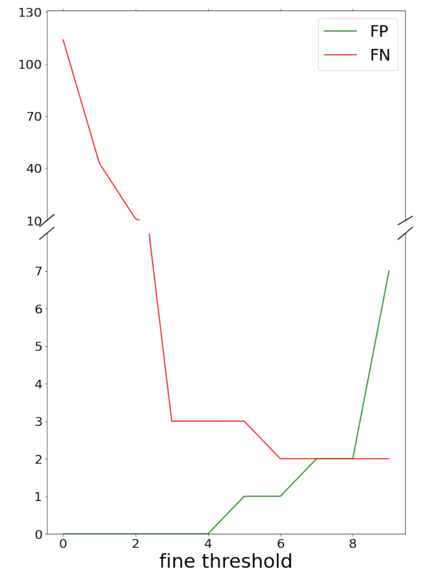


Figure 4.11. False positives and false negatives versus fine threshold (Tang et al., 2021)

In the second step of inspection, a fine threshold was applied to carry out inspection more rigorously and further filter out minor defects. Figure 4.11 presents the FP and FN with a fine threshold between 0 to 9. Choosing a fine threshold of four gave a FP of zero and a FN of three, which meant a 100% precision.

All the parameters of the inspection process of the first trail of dataset setting three are listed in Table 4.5.

Table 4.5. *Inspection parameters (Tang et al., 2021)*

	Parameters	Value
Step1	Coarse Threshold	31
	Window Size	3
Step2	Fine Threshold	4
	Gradient Threshold	70
	Window Size	5
	Gradient Operator	7x7

Table 4.6. *Performance of all trails under different settings (Tang et al., 2021)*

Setting	Train size	Trial	accuracy	precision	recall	TP	TN	FP	FN	coarse threshold	fine threshold
1	10	1	99.07%	100%	98.04 %	100	114	0	2	38	7
1	10	2	91.67%	96.67%	85.29 %	87	111	3	15	38	3
1	10	3	95.37%	100%	90.20 %	92	114	0	10	28	3
2	20	1	97.09%	100%	93.48 %	86	114	0	6	26	3
2	20	2	98.06%	100%	93.56 %	88	114	0	4	28	3
2	20	3	97.09%	100%	93.48 %	86	114	0	6	26	2
3	30	1	98.47%	100%	96.34 %	79	114	0	3	31	4
3	30	2	97.45%	100%	93.90 %	77	114	0	5	30	4
3	30	3	96.43%	100%	91.46 %	75	114	0	7	31	3
4	40	1	99.46%	100%	98.61 %	71	114	0	1	32	5
4	40	2	96.77%	100%	91.67 %	66	114	0	6	28	2
4	40	3	97.31%	100%	93.06 %	67	114	0	5	28	4

There were four dataset settings, under which the size of training set was 10, 20, 30, and 40, respectively. Three trials for each setting were conducted to study the robustness of the proposed method, and the dataset was randomly split in each trial. Therefore, there were a total of 12 experiments. Table 4.6 presents the performance and threshold setting of each trail. Only in the second trial of setting one, a proper fine threshold could not be found to keep the FP as zero while a low FN was maintained. All the trials of each experiment gave similar thresholds and similar performance, which suggested that the proposed method had a good repeatability rather than overfitting on a specific split. An 100% precision and a high recall were ensured in most trials. This suggested that the proposed method was unlikely to allow defective products to pass the inspection, and all defective products were rejected. Setting one yielded a relatively poor performance due to its extremely limited size of training set, while other settings yielded a similar performance. It should be noted that due to a relative small data size, a minor change to FP or FN could have a relative significant impact on precision, recall, and accuracy. The performance of the proposed method was reported by the first trial of setting three, as comprehensively discussed. The proposed method achieved an accuracy of 97.45%, a recall of 93.90%, and a precision of 100% using only a training set of 30 images.

4.4.7 Performance Comparisons

Template matching had already been discussed in Section 4.4.2, and it was concluded that template matching failed the inspection. The proposed algorithm was compared with two more popular conventional anomaly detection algorithms: the OC-SVM and Local Outlier Factor (LOF). These two algorithms were vector-based and were applied in conjunction with the CAE dimension reduction which transforms the image into vectors. The setting of automatic preprocessing and the CAE parameters were completely the same as the proposed method to ensure the fairness of comparison. The trained model from the first trial of setting three was used in all the comparative experiments. The embeddings of all the samples were centralized and normalized with the averaged values and variance of each dimension. Both the OC-SVM and LOF were implemented by Scikit-Learn 0.21.3 (Pedregosa et al., 2011). For OC-SVM, different parameter settings of kernel, degree, gamma coefficient, and coefficient-zero were tested and the corresponding results are presented in Table 4.7. For LOF, different parameter settings of base algorithm, number of neighbors, and leaf size were tested, and the corresponding results are presented in Table 4.8.

OC-SVM and LOF failed the inspection task completely. The major reason was that these two famous methods are developed for vector data, and they struggle with image data even after a dimensional reduction by CAE. Another reason was that the limited training dataset led the corresponding vector space to be highly sparse, thus causing overfitting for the algorithm. Similar to this study, Lehr et al. (2020) proposed to use CAE and difference image anomaly detection on industrial images as well. However, they only tested a fully synthetic dataset which had no noises like placement locations, orientations, and X-ray lights. Only a single threshold was applied on the difference image to obtain the final inspection performance. If their method was applied in a real-world scenario like this die casting core inspection, a low threshold would be selected to overcome the noises and ensure a high precision. This would cause a significant degeneration of recall.

Table 4.7. *CAE + OC-SVM performance (Tang et al., 2021)*

kernel	degree	coef0	precision	recall	accuracy
linear			30.61%	18.29%	48.47%
poly	3	0.0	37.21%	19.51%	52.55%
poly	3	1.0	64.86%	29.27%	63.78%
poly	5	0.0	41.67%	18.29%	55.10%
poly	5	1.0	52.17%	29.27.%	59.18%
poly	10	0.0	52.28%	34.15%	59.70%
poly	10	1.0	52.08%	30.49%	59.18%
rbf			52.69%	59.76%	60.71%
sigmoid		0.0	35.61%	57.31%	38.77%
sigmoid		1.0	33.73%	35.37%	43.88%

Note: The parameter setting of ‘gamma’ with either *scale* or *auto* was not presented since no impact was observed on all presented settings. The parameters that conflicted with the kernel setting were left blank.

Table 4.8. *CAE + LOF performance (Tang et al., 2021)*

n_neighbors	precision	recall	accuracy
5	43.78%	98.78%	46.43%
10	41.05%	95.78%	40.81%
20	41.75%	98.78%	41.81%

Note: The parameter setting of ‘algorithm’ including *auto*, *ball_tree*, *kd_tree* or *brute*, and the parameters setting of ‘leaf_size’ of *10,20,30* showed no impact on all presented settings.

4.4.8 Inference Speed

Since sliding window polices were applied to the neural network output, additional processing time was required. However, in our experiments, inferring each image took less than 0.5 seconds, and therefore, inferring speed was not the bottleneck here.

4.4.9 Generalizability Discussion

As a consequence of data availability, the proposed algorithm was only evaluated on the die casting core dataset. Nonetheless, the ideas of applying this algorithm on other datasets are shared. The main structure of the proposed algorithm, CAE, is intrinsically an dimension reduction algorithm. For that reason, it has limited ability to adapt the dataset with high variability, like cats images or dog images. On the contrary, the industrial inspection images of non-defective products are often of less variability, since they are all manufactured according to strict design standards. Most variations originate from the difference of positioning while the product is transmitted to the inspection system. Hence, CAE can obtain a considerable performance on these images with relatively limited data. Therefore, as long as the products have uniform geometry as designed, the proposed method should have no problem adapting to them. Rotation was the only augmentation in this study, which was because the area of interest can be easily located through Hough transformation in this case. It should be noted that selecting proper augmentation can be a key factor for training the CAE. The augmentation should correspond to the variation of the dataset, which was determined by the manufacturing process and inspection process. For instance, if the distance between the scanner and the product was inconsistent, the center crop within proper degree could be applied to simulate the distance variation. If the products were transmitted into the inspection device with variation in placement angle, random crop could be applied. Overall, augmentation strategies can be freely designed according to the specific inspection scenario.

The input images were reshaped to 128×128 in this study. However, the proposed approach was not limited to it. By adjusting the number of layers and stride of each layer, CAE could adapt to the images of any input size. However, it is recommended to limit the size of the image to the level on which humans can perform inspection, instead of using an overly large input size. This reduces the number of layers and that of parameters in the CAE, which reduces training effort and the chance of overfitting. The convolutional network could also be replaced with fully convolutional networks (Long, Shelhamer, & Darrell, 2015), which takes input with arbitrary size input and reduces the dimension with a preset ratio. It should be noted that the bottleneck of the model should not be overly loose to the extent that the network would cheat by copying the input to the output, and not be overly tight to the extent that the network fails in reconstruction. The procedure described in Section 4.4.1 serves as a great example. The size and threshold of the sliding window should also be adjusted according to the size of input images. One can follow the procedure detailed in Section 4.4.4 to decide the proper parameters.

CHAPTER 5. SEMI-SUPERVISED ANNOTATION

This chapter presents the experiments of the proposed semi-supervised annotation algorithm SemiCon for industrial images. The dataset and training parameters are introduced. Then, the performance of the model and a series of ablation studies are reported. At last, the generalizability of the proposed method is introduced.

5.1 Dataset and Hardware

Two datasets were selected for the research. One was the Northeastern University (NEU) steel surface defect dataset (Song & Yan, 2013), and the other was the concrete surface defect dataset (L. Zhang, Yang, Zhang, & Zhu, 2016). All of the experiments were completed on a workstation with an i9-10900KF CPU, an Nvidia 3090 GPU with 24 GB of memory, and a total of 32 GB DDR4 RAM using Pytorch 1.10, faiss-gpu 1.7, and Python 3.8.

The NEU steel surface defect dataset was comprised of the images of defective hot rolled steel surface strip. There are were total of six categories of defects, i.e., crazing (Cr), inclusion (In), patches (Pa), pitted surface (PS), rolled-in scale (RS), and scratches (Sc). Each category included 300 images. All of these images were in gray-scale and have a resolution of 200×200. Figure 5.1 presents four images of each category from this dataset. This dataset is referred to as NEU dataset in the following text.

The annotated dataset contained 30 randomly chosen images from each category ($N = 180$ images, 10% of the whole dataset), and the remaining 1620 images were assigned to the unannotated dataset.

The concrete surface defect dataset involved two categories, non-defective and defective. Each category had $20k$ images and there were $40k$ images in total. All images were colored and had a resolution of 227×227. Figure 5.2 presents eight images from each category. It was obvious that this dataset was much more easier than the NEU dataset. Therefore, 10 images from each category were randomly chosen to construct the annotated dataset ($N = 20$ images, 0.5% of the whole dataset), and the remaining 39980 images were assigned to the unannotated dataset.

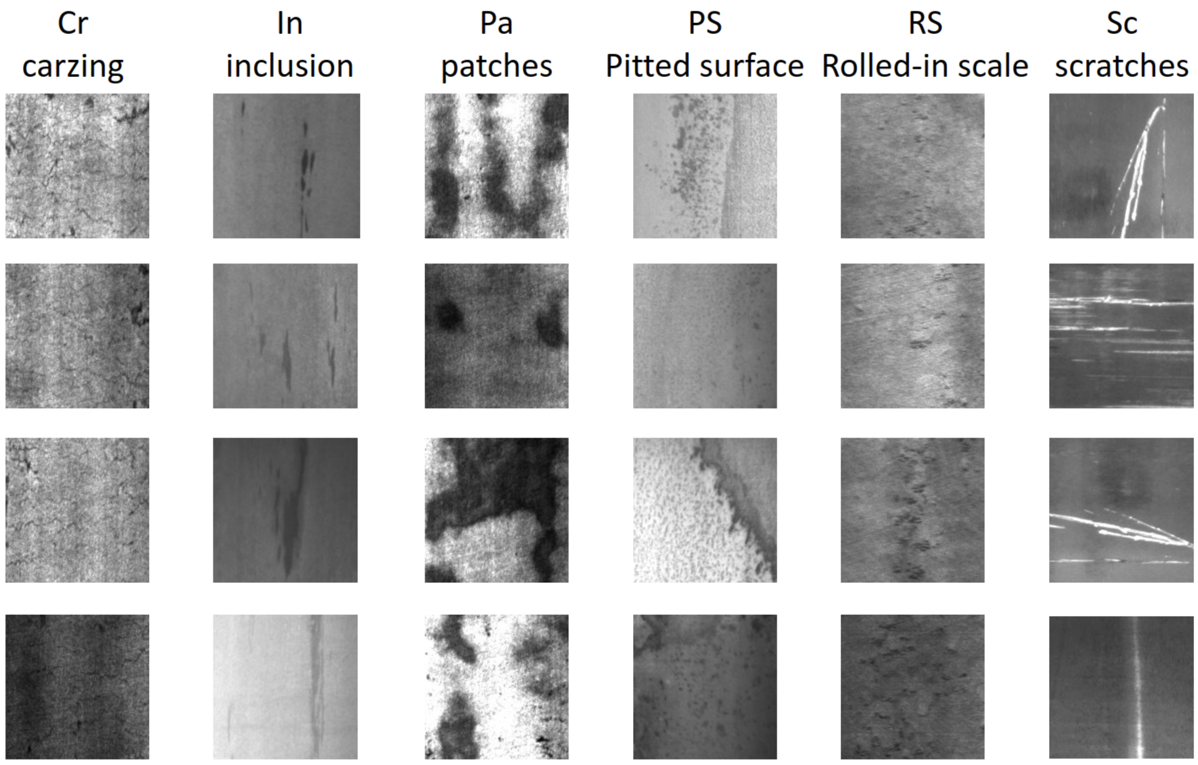


Figure 5.1. NEU dataset illustration.

5.2 Training parameters

The batch size of annotated images *m* was set to 32 and 20 in NEU dataset and concrete surface defect dataset, while the batch size of unannotated images *n* was set to 64 and 128 in NEU dataset and concrete surface defect dataset. The temperature parameter τ in loss function was set to 0.07 following Khosla et al. (2020), while α and *k* in vector based inference were set to 0.5 and 15 following Lazarou, Stathaki, and Avrithis (2021). The weak augmentation included horizontal flip and vertical flip with 50% probability, while the strong augmentation included additional randomly resized crop with a scale from 0.75 to 1. All images were resized to 112×112 before being inputted into the model. AutoAugment (Cubuk et al., 2019) was not applied as in Sohn et al. (2020), since validation set was not available in the setting. The model was trained using SGD optimizer with a momentum of 0.9 and a weight decay of 0.001 for 10*k* epochs. It should be noted that an epoch was defined as a full round of iterations on the annotated set, which was extremely small. More specifically, one epoch had only six iterations in NEU dataset

($round\_up(N\backslash n) = round\_up(180\backslash 32) = 6$) and as few as a single iteration in concrete surface defect dataset. The learning rate was set to 0.01 for the initial 100 epochs with warm-up for the first 10 epochs, and then it was adjusted to be 0.001 for resetting the training. As for confidence threshold and feature bank size, ablation studies were performed to evaluate their impact on model performance.

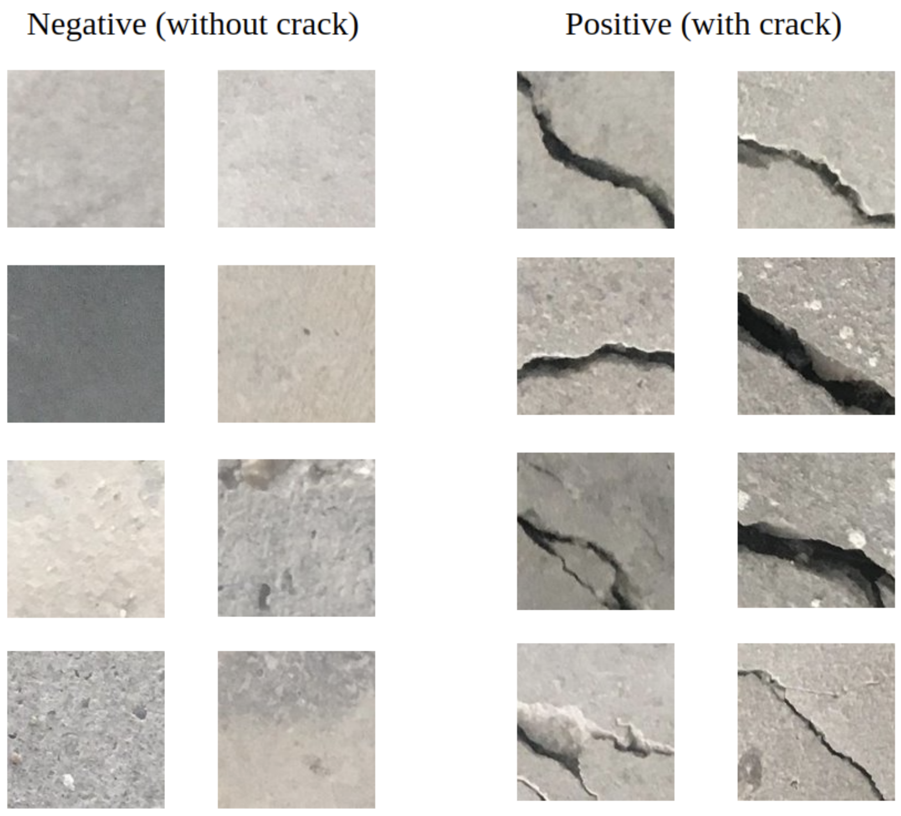


Figure 5.2. Concrete surface defect dataset illustration.

To fine-tune the classifier *Cla*, cross entropy loss and a SGD optimizer with a momentum of 0.9 and weight decay of 0 were used. The classifier was trained for 100 epochs in total. The first 30 epochs had a learning rate of 0.1, 31 to 60 epochs had a learning rate of 0.01, and 61 to 100 epochs had a learning rate of 0.001.

To ensure the repeatability of all the experiments, the random seeds of all experiments were set to 123. This ensured that running with the identical parameter gave complete identical results. Table 5.1 presents all of the hyperparameters excluding the confidence threshold and feature bank size.

Table 5.1. *Hyperparameters*

hyperparameter	notation	value
annotated dataset batch size	m	NEU 32 Concrete 20
unannotated dataset batch size	n	NEU 64 Concrete 128
temperature in loss function	τ	0.07
k in KNN	k	15
α in graph inference	α	0.5
momentum of momentum encoder	*moment*	0.999

Note: The confidence threshold and feature bank size are excluded.

5.3 Baseline and Benchmark

To set up competitors for the proposed method, a baseline and a benchmark were constructed. The baseline was to apply the conventional cross entropy (CE) classification on annotated dataset. This approach had been implemented in a series of industrial image classification studies including P.-H. Chen and Ho (2016); H. Lin et al. (2019); Masci et al. (2012); Özgenel and Sorguç (2018); Yi et al. (2017); L. Zhang et al. (2016). The baseline model architecture was set as ResNet-50, which was the same as SemiCon. The model was trained for 500 epochs using a SGD optimizer with a momentum of 0.9 and a weight decay of 0.001. The strong augmentation and input size as defined in Section 5.2 were used. The model was trained at a learning rate of 0.01 for the first 200 epochs and then the learning rate was reduced to 0.001 for the next 300 epochs. The batch size was set to 32 on NEU dataset and 20 on concrete surface defect dataset. The CE baseline achieved a 94.88% accuracy on the NEU unannotated dataset and a 94.82% accuracy on the concrete surface defect dataset.

Xie, Dai, et al. (2020) UDA concluded the innovations of previous research and proposed a simple but effective model. It was chosen as the benchmark to compete with SemiCon. The augmentation and batch sizes were the same as SemiCon, and the confidence threshold was set to 0.99 and 0.95. Additionally, the softmax temperature was 0.4 and the loss ratio was 1.0, both of which were the default values in UDA. The optimizer was the same as SemiCon. The model was trained for 1000 epochs. The initial learning rate was 0.01 and it dropped with a ratio of 0.1 at 100 and 500 epochs. The UDA achieved an accuracy of 95.80% with a confidence threshold of 0.99 and an accuracy of 94.94% with a confidence threshold of 0.95. The higher performance of the two experiments, an accuracy of 95.80%, was chosen as the performance of UDA. As for the concrete surface defect dataset, using a confidence threshold of 0.99 led to an accuracy of 97.48%, and it was chosen as the performance of UDA.

5.4 Experiments on the NEU dataset

This section presents all the experiments and studies on the NEU dataset. In all experiments, the performance was reported as the evaluation result of the last epoch, despite a better performance achieved during training.

5.4.1 Confidence Threshold

Table 5.2 presents the model performance with different confidence thresholds using a feature bank size of 300. The model achieved the highest performance and an accuracy of 97.41%, with the confidence threshold set to 0.95 and 0.90. Using a confidence threshold above 0.90 gave a higher performance and reducing it below 0.90 caused the performance to deteriorate significantly. This was because a too low confidence threshold introduced more false predictions among the pseudo labels, which affected the model performance negatively. Meanwhile, using a too high confidence threshold set a harsh criterion on pseudo labeling and reduced the number of pseudo labeled samples. It can be observed that the quality of pseudo labels was more significant to improving accuracy than the number of pseudo labels. The pseudo labeling process of using the confidence thresholds of 0.95 and 0.80 is compared in Figure 5.3. It can be observed that the top image has a higher blue line of total pseudo labels per epoch than the lower line. This suggests that using a lower confidence threshold of 0.80 introduces more pseudo labels than using a higher confidence threshold of 0.95. Meanwhile, the yellow line departs from the blue line more significantly in the top image, which suggests that the quality of the pseudo labels with a confidence threshold of 0.80 is lower than the quality of the pseudo labels with a confidence threshold of 0.95.

Table 5.2. *Confidence threshold*

algorithm	conf_thresh	accuracy
SemiCon	0.99	97.10%
SemiCon	0.95	97.41%
SemiCon	0.90	97.41%
SemiCon	0.85	95.93%
SemiCon	0.80	95.74%
base-line (CE)		94.88%
benchmark (UDA)	0.99	95.80%

Note: The feature bank size was set to 300, and other parameters were set as introduced in Section 5.2. Baseline and benchmark were included for comparison.

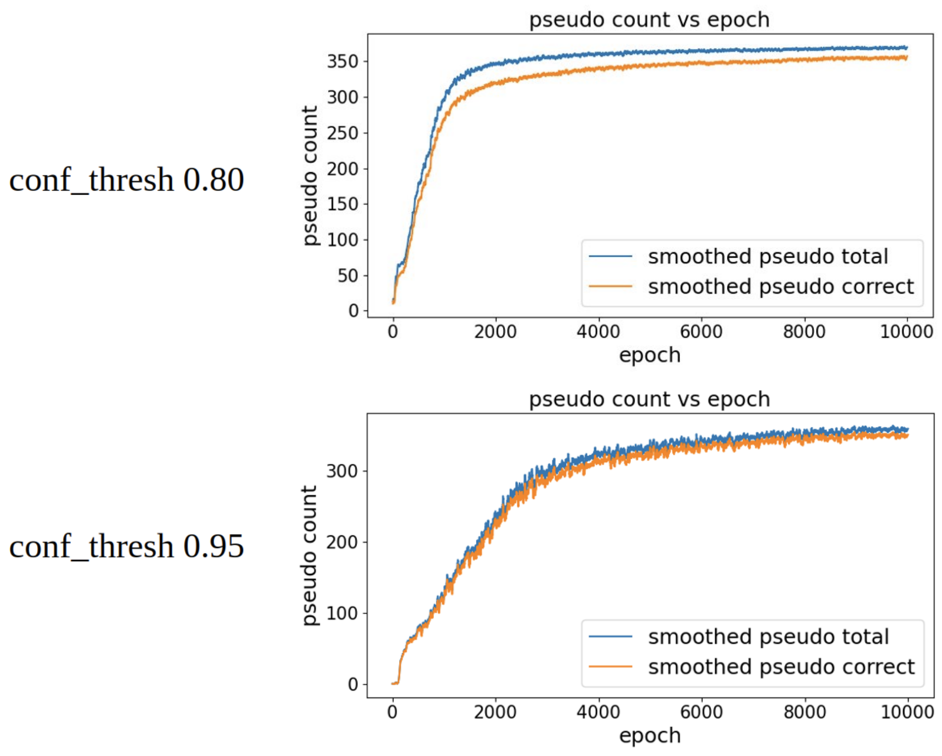


Figure 5.3. Pseudo labeling of experiments

5.4.2 Feature Bank Size

Table 5.3 presents the model performance with different feature bank sizes. Using a feature bank size of 300, 900, and 1500 gave an accuracy of 97.41%, 97.78%, and 97.78%. It is believed that a larger bank size provides more delicate references during vector bank inference and contributes to the performance of the model. However, in the experiments, the performance gained from increasing the size of feature bank from 300 to 900 was insignificant, and there was even no performance gain from increasing the size of feature bank from 900 to 1500. A larger bank size than 1500 was not tested, since 1500 was almost as large as the size of the whole unannotated dataset.

Table 5.3. *Feature bank size*

method	bank size	accuracy
SemiCon	300	97.41%
SemiCon	900	97.78%
SemiCon	1500	97.78%
base-line (CE)		94.88%
benchmark (UDA)		95.80%

Note: The confidence threshold was set to 0.95, and other parameters were set as introduced in Section 5.2. Baseline and benchmark were included for comparison.

5.4.3 Best Combination

Combining the experiments in Section 5.4.1 and Section 5.4.2 gave the best hyper parameter combination. According to Section 5.4.1, a confidence threshold of 0.90 or 0.95 was chosen, since they were neither too low to introduce more false pseudo labels nor too high to limit the amount of pseudo labels. According to Section 5.4.2, a larger size of feature bank benefits the model performance and the feature bank size was set to 1500. Using a confidence threshold of 0.95 and feature bank size of 1500 achieved an accuracy of 97.78%, as shown in Table 5.3. Using a confidence threshold of 0.90 and feature bank size of 1500 achieved an accuracy of 97.90%, which was the best model performance in all experiments. The best performance of SemiCon, baseline, and benchmark is presented in Table 5.4.

Table 5.4. *Best model performance*

method	bank size	conf_thresh	accuracy
SemiCon	1500	0.90	97.90%
base-line (CE)			94.88%
benchmark (UDA)		0.99	95.80%

5.4.4 KNN vector based inference

As mentioned in Section 3.2.5, a naive solution to vector-based inference was KNN. The weighted KNN inference with the number of neighbors set to 20 and euclidean distance was chosen to draw comparison with the proposed graph-based inference. The feature bank size and confidence threshold were set to 300 and 0.90. This setting achieved an accuracy of 96.11%, which was significantly lower than when the proposed graph-based inference method was used with other parameters unchanged. This pair of experiments were concluded in Table 5.5. The pseudo labels during the training of these two experiments were compared, as shown in Figure 5.4. It can be observed that the proposed graph-based inference obtained the pseudo labels with a higher accuracy than weighted KNN. The use of weighted KNN with a higher confidence threshold of 0.95 was also tested. However, the training failed since weighted KNN obtained a very small number of pseudo labels given such high confidence threshold.

Table 5.5. *Weighted knn inference*

method	vector based inference method	accuracy
SemiCon	(Zhou et al., 2003)	97.41%
SemiCon	weighted knn	96.11%
base-line (CE)		94.88%
benchmark (UDA)		95.80%

Note: The confidence threshold was set to 0.90, feature bank size was set to 300, and other parameters were set as introduced in Section 5.2. Baseline and benchmark were included for comparison.

5.5 SemiCon Training Process Analysis

There were a series of interesting characteristics of the training process observed during training. To illustrate these characteristics, the training record of two representative experiments was presented. The first experiment (a) was the first setting in Table 5.3 with a confidence threshold of 0.95 and a feature bank size of 300, while the second experiment (b) was the best performance experiment in research as introduced in Section 5.4.3, with a confidence threshold of 0.90 and a feature bank size of 1500.

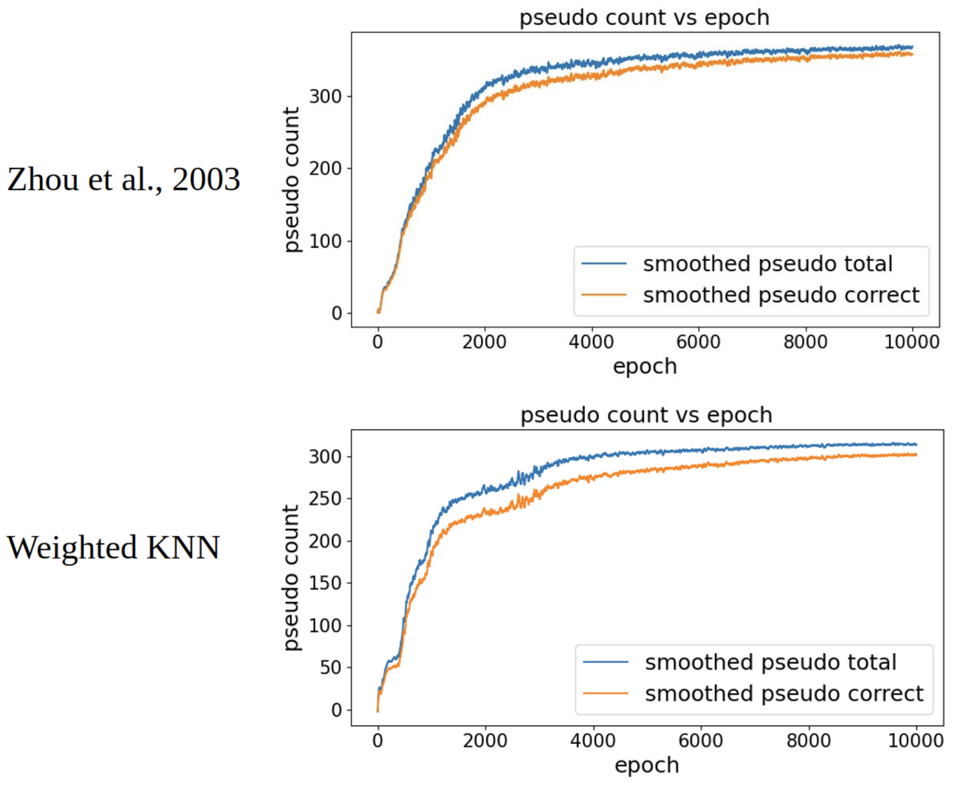


Figure 5.4. Pseudo labeling of weighted KNN

Figure 5.5 presents the training records of the two experiments, with column (a) for experiment (a) and column (b) for experiment (b). The upper row is the training loss versus epoch number, the middle row is the accuracy on the unannotated dataset versus epoch number, and the bottom row is the total pseudo label at each epoch and correct pseudo label at each epoch. For experiment (a), it was quite unique that the training loss was not reduced throughout the training as presented in the top row of column (a). Instead, it dropped during the first 1000 epochs, and then gradually increased to a steady state. Despite the increase of training loss, the accuracy on unannotated set kept increasing until reaching a steady state. The reason for this was that in the initial epochs, only a small number of unannotated images were pseudo labeled. The number of pseudo labeled images can be found in column (a) bottom image. Therefore, the actual training set was limited and there were relatively insignificant variations, which made it easy to perform training task. As training proceeded, more and more images were pseudo labeled, thus making the training task harder and harder to perform, and the loss increased gradually. The final equilibrium was reached when the number of pseudo labeled per epoch ceased to increase, and almost all images were pseudo labeled. It should be noted that there was an upper limit on the total pseudo labeled images. In this experimental setting, each epoch had six iterations, which meant a total of $n \times 6 = 384$ images from the unannotated dataset.

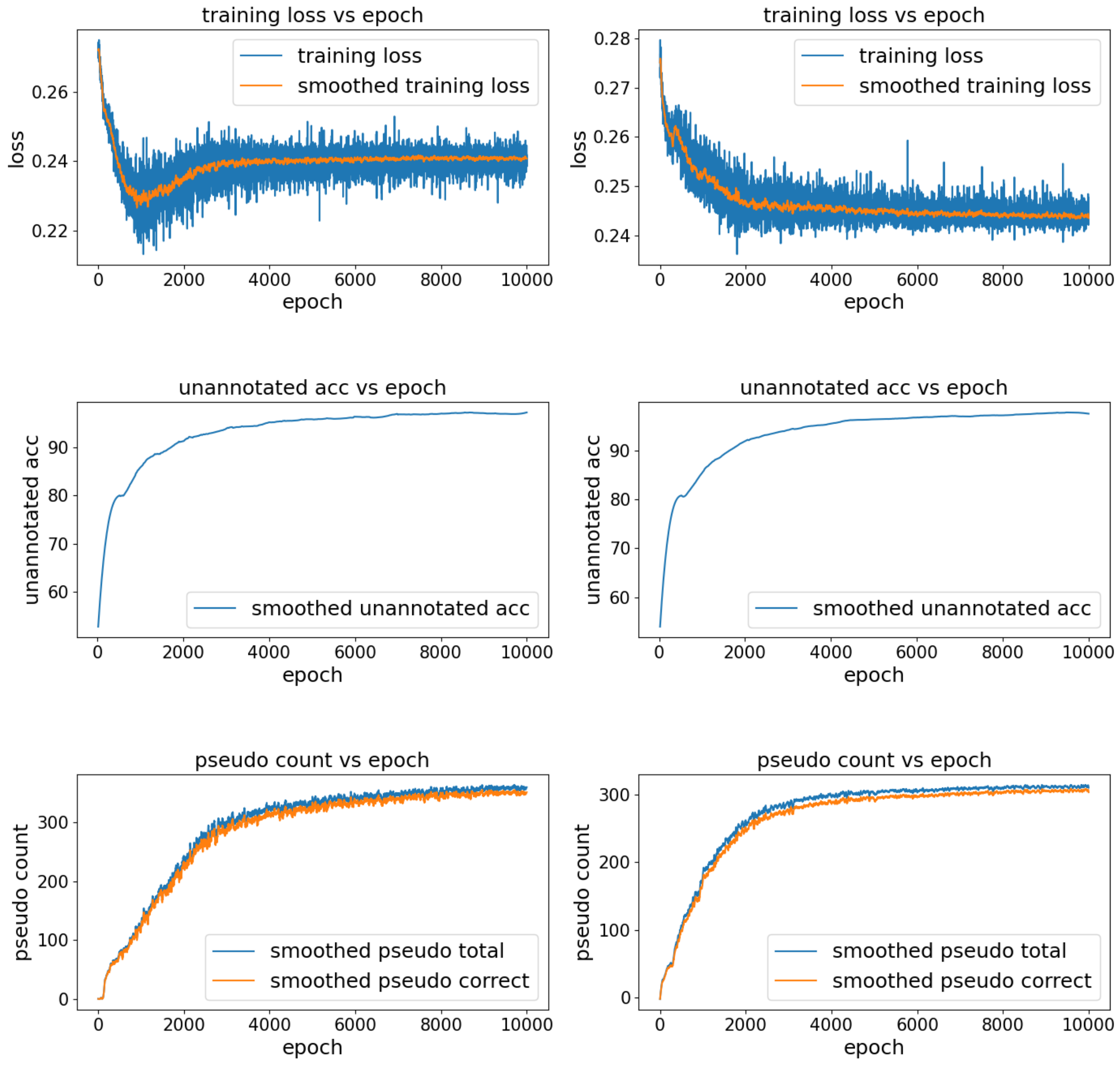


Figure 5.5. Training analysis of SemiCon

This reversal of training loss was barely observed in the second experiment with a confidence threshold of 0.90 and a feature bank size of 1500, as presented in column (b) of Figure

5.5. Compared with the first experiment, it had a lower confidence threshold but a larger feature bank. This caused the model to have more pseudo labels and a larger training set, as can be found by comparing the bottom image of column (a) and column (b). The introduction of more images into the training process increase the variation of the training set. This led to a more regular training loss that dropped consistently, except for the very initial stage (around 400 epochs) when the number of pseudo labels drastically increased. An interesting fact was that except the different behaviours of the training loss in two experiments, they both converged to a similar value of 0.24.

What is also noteworthy is that SemiCon took a relatively longer time to train. In the introduced workstation, SemiCon took approximated 6 hours to converge, while the benchmark UDA took approximately half an hour to do so. A key factor was that momentum encoder restricts the updating speed of the model, which is aimed in design to improve the uniformity of the feature encoder and the quality of the pseudo labels. However, SemiCon, benchmark UDA, and baseline CE are equivalent in terms of inference speed.

5.5.1 UDA Training Process Analysis

To better compare the proposed contrastive learning based SemiCon with conventional cross entropy based UDA, Figure 5.6 presents the training process of UDA with a confidence threshold of 0.99. As can be seen in the upper image, the training loss of UDA drops drastically and converges after only 200 epochs. This suggests that the training of UDA was far more easier than SemiCon. It is reasonable since cross entropy training is much more straightforward than contrastive learning. In cross entropy training, the model directly minimizes the distance between the feature vectors of samples and their corresponding category centers, while in contrastive learning training, the model optimizes the distance between the feature vectors of all samples. More specifically, it is assumed that there are K categories, each category has N samples, and cross entropy optimizes the distance of $K \times N$ pairs (each sample to its category center). Meanwhile, contrastive learning optimizes the distance of $(K \times N)^2$ (each sample to each sample) pairs.

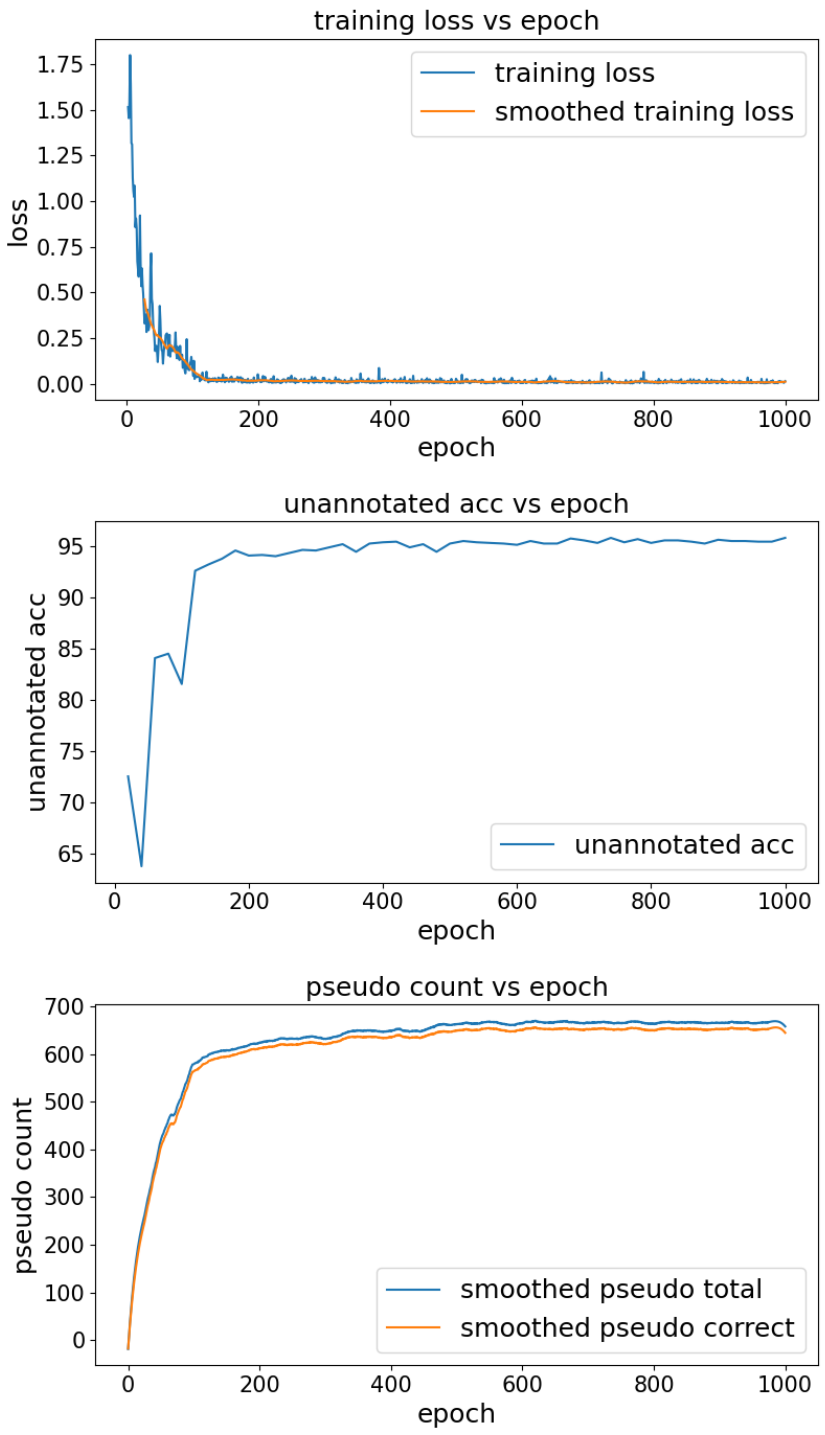


Figure 5.6. Training analysis of UDA with confidence threshold 0.99.

5.6 Experiments on Concrete Surface Defect dataset

This section presents all experiments conducted on the concrete surface defect dataset. Unlike on NEU dataset, SemiCon outperformed both baseline and benchmark, while SemiCon failed to surpass the benchmark. All of the experiments were concluded in Table 5.6.

Table 5.6. *Concrete surface defect dataset experiments*

algorithm	conf_thresh	feature bank size	accuracy
SemiCon	0.99	300	97.10%
SemiCon	0.95	300	96.25%
SemiCon	0.90	300	96.10%
SemiCon	0.99	3000	96.78%
SemiCon	0.95	3000	96.27%
SemiCon	0.90	3000	95.89%
base-line (CE)			94.82%
benchmark (UDA)	0.99		97.48%

Note: Other parameters were set as introduced in Section 5.2. Baseline and benchmark were included for comparison.

Despite no clear idea of why SemiCon failed to surpass the benchmark on the concrete surface dataset, there are several facts worth noting. Firstly, the concrete surface dataset is a binary classification dataset, which is rare in current research. In conventional unsupervised self-contrastive learning, since no label is available, each image is treated as an individual category, as a result of which the number of categories equals the dataset size. In supervised contrastive learning Khosla et al. (2020), the experiment datasets CIFAR-10, CIFAR-100, and ImageNet have 10, 100, and 1000 classes correspondingly. Therefore, to the best of our knowledge, there has never been research of contrastive learning on binary classification. Secondly, it is widely known that the performance of contrastive learning scales with the batch size. A larger batch size means a better performance, whether it is unsupervised self contrastive learning or supervised contrastive learning. However, in the context of auto annotation, the number of images in the training set is extremely small, which is adverse to enlarging the batch size from the initial of the training. Although the batch size is increased by pseudo labeled samples after initial training, this could still affect the performance.

5.7 Generalizability Discussion

The proposed algorithm named SemiCon makes a few assumptions of the dataset, and it can be easily adapted to any dataset like conventional semi-supervised classification algorithms. As introduced in Section 2.3, the two key factors for semi-supervised classification algorithms are augmentation consistency and pseudo labeling. The industrial inspection images are often subject to high constrains due the design of the products, which makes the selection of augmentation significant. This is also discussed in Section 4.4.9. In the experiments described in Section 5.4.1, using a high confidence threshold would not affect the model performance significantly, while using a low confidence threshold introduces false training data and reduces the model performance much more significantly. It is widely accepted in semi-supervised classification research that a higher confidence threshold is recommended for an easier dataset to ensure the quality of the pseudo label. Since industrial inspection images are more constrained and have less significant variations than those common computer vision images like cats and dogs, a confidence threshold of at least 0.9 was recommended. The experiments from Section 5.4.2 showed that using a large feature bank size led to a better model performance, or at least did not affect the model performance. Therefore, it is recommended to choose a large feature bank size within the limit of the work station GPU memory and also smaller than the unannotated dataset.

Another point worth noting is that the proposed algorithm fails to surpass the benchmarks in binary classification, but the reason for this remains unclear. Therefore, it is recommended to use conventional cross entropy base algorithm like UDA in the binary classification scenario.

CHAPTER 6. SUMMARY AND FUTURE RESEARCH

This chapter concludes the whole research, and discusses the future research direction.

In this research, the use of unsupervised learning and semi-supervised learning in the inspection of industrial images was investigated. Two specific studies, the unsupervised anomaly detection and semi-supervised annotation of industrial images, were performed. In unsupervised anomaly detection, a two-stage inspection algorithm using convolutional autoencoder was proposed. On a dataset provided by a die-casting foundry, it achieved an impressively high accuracy of 97.45% using only 30 good images. In the semi-supervised annotation of industrial images, a new algorithm named SemiCon was proposed. SemiCon introduces contrastive learning into semi-supervised classification to improve performance. The algorithm was tested on two public datasets: the NEU steel surface defect dataset and the concrete surface defect dataset. On the NEU dataset, SemiCon achieved an accuracy of 97.90% using only 10% annotation of all data, which is 3.02% higher than the baseline and 2.10% higher than the benchmark. On the concrete surface defect dataset, SemiCon achieved an accuracy of 97.10% using only 0.05% annotation of all data, which surpassed the baseline by 2.28% but fell short of the benchmark by 0.38%. In general, this research proved the effectiveness and potential of unsupervised learning and semi-supervised learning in automatic industrial image inspection. This is vital because in real-world scenarios, the annotation of training data in large amounts can be overly labor-intensive and not flexible enough for constantly updating manufacturing scenarios.

Despite the high performance of the proposed methods, their application scenario should be carefully evaluated. The proposed CAE anomaly detection algorithm is only capable of distinguishing those defective images with good images, and it cannot classify the different types of defects. Also, since CAE is a dimensional reduction algorithm, the information loss occurring during the compression would cause a slight variation between the input image and output image. This variation can be mistaken for the blurring defects like porosity, so that the CAE anomaly detection is incapable of detecting these defects. The CAE anomaly detection is also not capable of measuring those detailed parameters like diameter. In such scenario, the image processing based algorithms are preferred since detailed parameters can be measured. The proposed SemiCon and other semi-supervised classification algorithms are capable of both annotating unannotated images and training a classifier for the inspection. However, since inspection scenario requires a high performance, an additional annotated validation set is required to measure the performance while training a classifier. If the model performance did not meet the performance requirement, more data should be fed into the model until the performance requirement is met.

There are several directions for the future research on both unsupervised anomaly detection and semi-supervised annotation. For unsupervised anomaly detection, a promising research direction is to incorporate GAN to enlarge the training dataset and to use the discriminator of GAN as the inspection algorithm. Another promising research direction is to apply a clustering algorithm to separate different sources of defect, since the original anomaly detection algorithm only detects defective images, and it is not capable of distinguish between different types of defects. This can provide more information on the source of defects, which helps the engineers to better optimize the manufacturing process. For semi-supervised annotation, the first direction is the impact of unbalanced dataset. Unbalanced dataset has a significant impact on model performance and is common in real-word inspection scenarios. A conventional approach is to balance the training classes at each iteration. Another approach that is worth investigating is to first estimate the distribution of classes by randomly annotated images, and then optimize the vector based inference algorithm with an additional $Sinkhorn-Knopp$ algorithm. The second direction is to identify novel classes. In real-word scenarios, there is no guarantee that all defects are in the preset categories, and a new type of defects can appear during production. Novel class detection algorithms are the good sources of ideas. Other domains like the semi-supervised object detection and semi-supervised segmentation to identify and localize defects are also promising in automatic industrial inspection.

REFERENCES

Benmoussat, M., Guillaume, M., Caulier, Y., & Spinnler, K. (2013). Automatic metal parts inspection: Use of thermographic images and anomaly detection algorithms. *Infrared Physics & Technology*, *61*, 68–80.

Bian, X., Lim, S. N., & Zhou, N. (2016). Multiscale fully convolutional network with application to industrial inspection. In *2016 ieee winter conference on applications of computer vision (wacv)* (pp. 1–8).

Cao, K., Wei, C., Gaidon, A., Arechiga, N., & Ma, T. (2019). Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, *32*.

Chen, P.-H., & Ho, S.-S. (2016). Is overfeat useful for image-based surface defect classification tasks? In *2016 ieee international conference on image processing (icip)* (pp. 749–753).

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *International conference on machine learning* (pp. 1597–1607).

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 113–123).

Davies, E. (1984). Circularity—a new principle underlying the design of accurate edge orientation operators. *Image and Vision Computing*, *2*(3), 134–142.

Davies, E. R. (2012). *Computer and machine vision: theory, algorithms, practicalities*. Academic Press.

Demant, C., Garnica, C., & Streicher-Abel, B. (2013). *Industrial image processing*. Springer.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

Erhan, D., Courville, A., Bengio, Y., & Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 201–208).

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*(2), 303–338.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 9729–9738).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hu, G., Huang, J., Wang, Q., Li, J., Xu, Z., & Huang, X. (2020). Unsupervised fabric defect detection based on a deep convolutional generative adversarial network. *Textile Research Journal*, *90*(3-4), 247–270.

Hyun, M., Jeong, J., & Kwak, N. (2020). Class-imbalanced semi-supervised learning. *arXiv preprint arXiv:2002.06815*.

Jin, B., Tan, Y., Nettekoven, A., Chen, Y., Topcu, U., Yue, Y., & Vincentelli, A. S. (2019). An encoder-decoder based approach for anomaly detection with application in additive manufacturing. *arXiv preprint arXiv:1907.11778*.

Kang, C. W., Ramzan, M. B., Sarkar, B., & Imran, M. (2018). Effect of inspection performance in smart manufacturing system based on human quality control system. *The International Journal of Advanced Manufacturing Technology*, *94*(9), 4351–4364.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., … Krishnan, D. (2020). Supervised contrastive learning. *Advances in Neural Information Processing Systems*, *33*, 18661–18673.

Kim, J., Hur, Y., Park, S., Yang, E., Hwang, S. J., & Shin, J. (2020). Distribution aligning refinery of pseudo-label for imbalanced semi-supervised learning. *arXiv preprint arXiv:2007.08844*.

Kimura, M., & Yanagihara, T. (2018). Anomaly detection using gans for visual inspection in noisy training data. In *Asian conference on computer vision* (pp. 373–385).

Knight, P. A. (2008). The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, *30*(1), 261–275.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097–1105.

Kuo, C.-F. J., Hsu, C.-T. M., Liu, Z.-X., & Wu, H.-C. (2014). Automatic inspection system of led chip using two-stages back-propagation neural network. *Journal of Intelligent Manufacturing*, *25*(6), 1235–1243.

Laine, S., & Aila, T. (2016). Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*.

Lazarou, M., Stathaki, T., & Avrithis, Y. (2021). Iterative label cleaning for transductive and semi-supervised few-shot learning. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 8751–8760).

Leavers, V. F. (1992). *Shape detection in computer vision using the hough transform* (Vol. 1). Springer.

Lee, D.-H., et al. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, icml* (Vol. 3, p. 896).

Lehr, J., Sargsyan, A., Pape, M., Philipps, J., & Krüger, J. (2020). Automated optical inspection using anomaly detection and unsupervised defect clustering. In *2020 25th ieee international conference on emerging technologies and factory automation (etfa)* (Vol. 1, pp. 1235–1238).

Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, *8*, 193907–193934.

Li, J., Xu, X., Gao, L., Wang, Z., & Shao, J. (2020). Cognitive visual anomaly detection with constrained latent representations for industrial inspection robot. *Applied Soft Computing*, *95*, 106539.

Lin, H., Li, B., Wang, X., Shu, Y., & Niu, S. (2019). Automated defect inspection of led chip using deep convolutional neural network. *Journal of Intelligent Manufacturing*, *30*(6), 2525–2534.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., … Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).

Liu, K., Li, A., Wen, X., Chen, H., & Yang, P. (2019). Steel surface defect detection using gan and one-class classifier. In *2019 25th international conference on automation and computing (icac)* (pp. 1–6).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 3431–3440).

Masci, J., Meier, U., Ciresan, D., Schmidhuber, J., & Fricout, G. (2012). Steel defect classification with max-pooling convolutional neural networks. In *The 2012 international joint conference on neural networks (ijcnn)* (pp. 1–6).

Mery, D. (2020). Aluminum casting inspection using deep learning: A method based on convolutional neural networks. *Journal of Nondestructive Evaluation*, *39*(1), 12.

Nar, K., Ocal, O., Sastry, S. S., & Ramchandran, K. (2019). Cross-entropy loss and low-rank features have responsibility for adversarial examples. *arXiv preprint arXiv:1901.08360*.

Niu, C., Shan, H., & Wang, G. (2021). Spice: Semantic pseudo-labeling for image clustering. *arXiv preprint arXiv:2103.09382*.

Oliveira, H., & Correia, P. L. (2014). Crackit — an image processing toolbox for crack detection and characterization. In *2014 ieee international conference on image processing (icip)* (p. 798-802). doi: 10.1109/ICIP.2014.7025160

Özgenel, Ç. F., & Sorguç, A. G. (2018). Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *Isarc. proceedings of the international symposium on automation and robotics in construction* (Vol. 35, pp. 1–8).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Racki, D., Tomazevic, D., & Skocaj, D. (2018). A compact convolutional neural network for textured surface anomaly detection. In *2018 ieee winter conference on applications of computer vision (wacv)* (pp. 1331–1339).

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rasmus, A., Valpola, H., Honkala, M., Berglund, M., & Raiko, T. (2015). Semi-supervised learning with ladder networks. *arXiv preprint arXiv:1507.02672*.

Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. *Advances in neural information processing systems*, *29*, 1163–1171.

Silvén, O., Niskanen, M., & Kauppinen, H. (2003). Wood inspection with non-supervised clustering. *Machine Vision and Applications*, *13*(5-6), 275–285.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., … Raffel, C. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*.

Song, K., & Yan, Y. (2013). A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects. *Applied Surface Science*, *285*, 858–864.

Staar, B., Lütjen, M., & Freitag, M. (2019). Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, *79*, 484–489.

Sukhbaatar, S., Bruna, J., Paluri, M., Bourdev, L., & Fergus, R. (2014). Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Tabernik, D., Šela, S., Skvarč, J., & Skočaj, D. (2020). Segmentation-based deep-learning approach for surface-defect detection. *Journal of Intelligent Manufacturing*, *31*(3), 759–776.

Tang, W., Vian, C. M., Tang, Z., & Yang, B. (2021). Anomaly detection of core failures in die casting x-ray inspection images using a convolutional autoencoder. *Machine Vision and Applications*, *32*(4), 1–17.

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*.

Thomas, A. D., Rodd, M. G., Holt, J. D., & Neill, C. (1995). Real-time industrial visual inspection: A review. *Real-Time Imaging*, *1*(2), 139–158.

Valpola, H. (2015). From neural pca to deep unsupervised learning. In *Advances in independent component analysis and learning machines* (pp. 143–171). Elsevier.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., & Van Gool, L. (2020). Scan: Learning to classify images without labels. In *European conference on computer vision* (pp. 268–285).

Wei, C., Sohn, K., Mellina, C., Yuille, A., & Yang, F. (2021). Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10857–10866).

Wieler, M., & Hahn, T. (2007). Weakly supervised learning for industrial optical inspection. In *Dagm symposium in.*

Xie, Q., Dai, Z., Hovy, E., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, *33*, 6256–6268.

Xie, Q., Luong, M.-T., Hovy, E., & Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10687–10698).

Yang, Y., & Xu, Z. (2020). Rethinking the value of labels for improving class-imbalanced learning. *arXiv preprint arXiv:2006.07529*.

Yi, L., Li, G., & Jiang, M. (2017). An end-to-end steel strip surface defects recognition system based on convolutional neural networks. *steel research international*, *88*(2), 1600068.

Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. In *2016 ieee international conference on image processing (icip)* (pp. 3708–3712).

Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, *31*.

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Schölkopf, B. (2003). Learning with local and global consistency. *Advances in neural information processing systems*, *16*.