Monitoring of Manufacturing process using Deep Learning

Project report submitted in partial fulfilment of the

requirements for the degree of

BACHELOR OF TECHNOLOGY

in

Mechanical Engineering

by

Keerthana S

(Roll No. 210103065)

Under the supervision of

Dr. Sukhomay Pal

to

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI – 781039, INDIA

[Jan-Apr 2025]

CERTIFICATE

This is to certify that the work contained in this project report entitled **“Monitoring of Manufacturing process using Deep Learning”** submitted by **Keerthana S (210103065)** to the Indian Institute of Technology Guwahati towards the partial requirement of **Bachelor of Technology** in **Mechanical Engineering** is a bona fide work carried out by her under my supervision and that it has not been submitted elsewhere for the award of any degree.

Dr. Sukhomay Pal

Mechanical Engineering

IIT GUWAHATI

Apr 2025

DECLARATION

I declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included, I have adequately cited and referenced the sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated, or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Keerthana S

Roll No. 210103065

Department of Mechanical Engineering,

I.I.T Guwahati, Assam, India

Approval Sheet

This project report entitled “Monitoring of Manufacturing Process using Deep Learning” by Keerthana S (210103065) is approved for the degree of Bachelor of Technology.

Dr. Sukhomay Pal

Mechanical Engineering

IIT GUWAHATI

Apr 2025

ACKNOWLEDGMENT

I would like to express my deepest gratitude to our supervisor **Prof. Sukhomay Pal**, Department of Mechanical Engineering, IIT Guwahati for their expert advice, valuable insights about the subject matter and exquisite support and encouragement throughout the course of my Bachelor of Technology titled "Monitoring of Manufacturing Process using Deep Learning". His methods, vision and enthusiasm has always been a constant motivation and inspiration to me.

Keerthana S

Roll No. 210103065

Department of Mechanical Engineering,

I.I.T Guwahati, Assam, India

CONTENTS

Chapter 1

Introduction

1.1 Abstract

This project is dedicated to the implementation of a model based on deep learning for detecting manufacturing defects via ultrasonic testing (UT) images. As a starting point, pre-trained models like VGG19 and ResNet50 were tested. Although ResNet50 had impressive performance during training, it didn't generalize as well to validation data. In contrast, VGG19 indicated impressive training accuracy and superior generalization properties albeit with significant volatility in validation performance. In light of the limitations encountered with pre-trained models, a user-defined convolutional neural network was established, optimized for the UT dataset of 256×256 grayscale images. The network structure consisted of building blocks such as max pooling, separable convolution layers, batch normalization, and dense layers, culminating in a final sigmoid-activated output appropriate for binary classification. The model was trained using a large set of training files and a small validation set, with 500 steps per epoch over 100 epochs. During training, the primary performance indicators, i.e., accuracy, recall, and AUC, showed consistent and dependable learning behavior. The custom model eventually offered more stable and generalizable performance than the original pre-trained models, rendering it more appropriate for real-world defect detection applications.

1.2 Motivation

Accurately detecting manufacturing defects is essential in many fields, including the automotive industry, electronics manufacturing, aerospace, and mechanical engineering. Early detection of manufacturing defects is particularly crucial, as it can prevent the production of faulty items, reduce the number of defective products, and result in significant cost savings and improved profitability. However, accurately predicting manufacturing defects is a challenging task that requires the use of advanced techniques and models. This project aims to develop a state-of-the-art deep-learning model for detecting manufacturing defects using ultrasonic images. By improving our ability to predict manufacturing defects, we can increase the safety in the automotive and aerospace industries, increase the reliability of electronic devices, improve efficiency in manufacturing processes, and prevent system failures in the aerospace and mechanical industries.

Chapter 2

Review

2.1 Literature Review

This research[1] elaborates on the function of deep learning models in sustainable manufacturing and proposes a framework to achieve it. The review indicates that deep learning research for manufacturing is scanty, and there is no awareness of its benefits in industries. The research illustrates how deep learning enhances quality management, predictive maintenance, reliability analysis, and fault diagnosis in sustainable production. The suggested framework can assist manufacturing companies in ensuring sustainable production in Industry 4.0 business models, with advantages including the minimization of machine downtime and enhanced fault diagnosis. Empirical testing of the suggested framework in different industries can be carried out in future studies.

This study[2] presents a novel approach utilizing big data and an AI-based decision-making tool for process optimization in manufacturing, a valuable contribution to Industry 4.0. The model, which has been trained on images of acceptable and faulty products, is as accurate as 97% in identifying defects, highlighting its potential in real-world production environments. The deployment of Industry 4.0 enablers such as machine learning, Cyber-Physical Systems (CPS), and the Internet of Things (IoT) is one of the main goals for increasing efficiency at the iFactory plant. The suggested model is made up of three stages that include dataset extraction from a supercomputer, pre-training with ResNet-50 for initializing the weights of the model, and end-to-end training to extract fine details and discriminative patterns. The structure of the model, shown in Fig. 9, is made up of a convolution layer, max-pooling layer, ReLU activation function, and SoftMax nonlinearity activation function with binary cross-entropy loss. The ADAM optimization technique further improves the performance of the model.

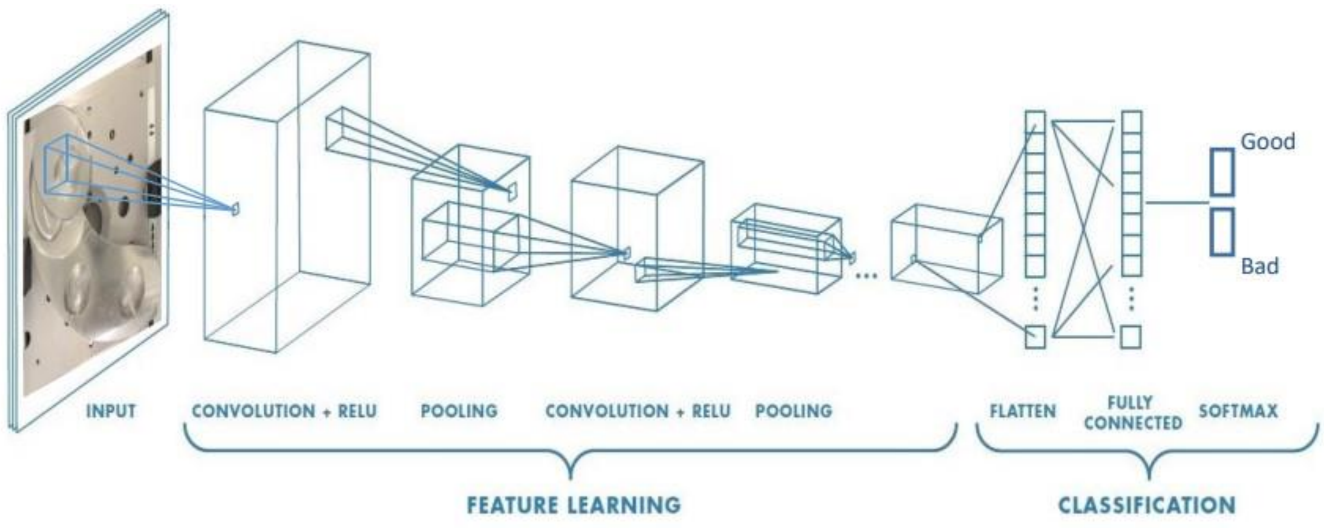

Figure 2.1.1 – Convolutional Network (Taken from [2])

Three current best-practice techniques for anomaly detection in images were compared on an eight-fold real-world ultrasonic NDT dataset in this work[3]. Improvements proposed were cropping-out patches from images for Ganomaly and applying a faster MobileNet network for PaDiM. PaDiM performed better than the other two techniques by detecting all defects in the dataset and outperforming classifiers when the ratio of positive samples in the training set was extremely small. The research points out the necessity of testing anomaly detection techniques against real-world data sets prior to applying them in production.

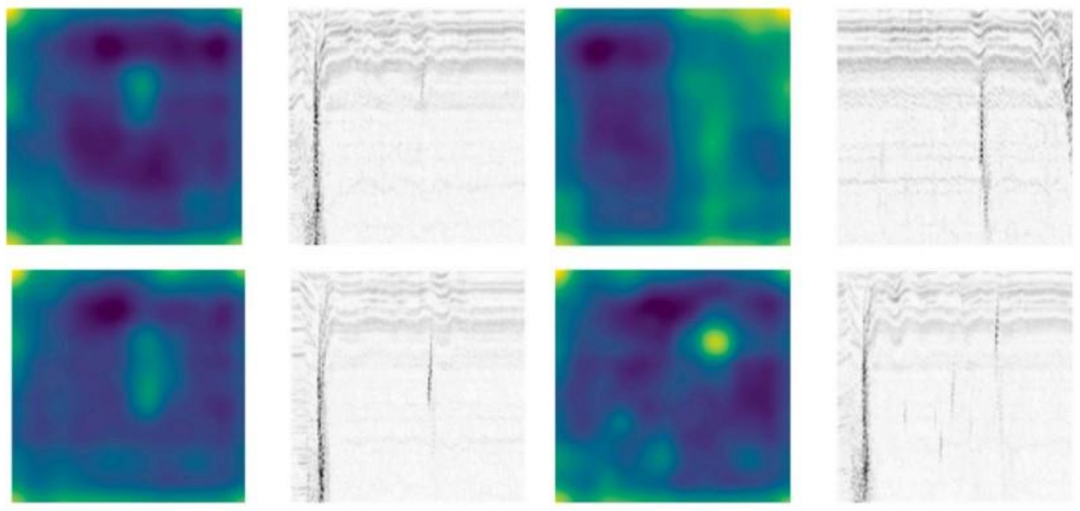

Figure 2.1.2 – Examples of anomalous images and attention from the PaDiM model(taken from [3])

The research[4] shows that the EfficientDet-D0 architecture is able to detect faults from images obtained with a phased-array probe. The authors optimized the performance of the network through a new method to calculate anchors' hyperparameters. The presented EfficientDet-D0 model showed an mAP of 89.6%, outperforming YOLOv3 by 9%. Nonetheless, in order to check its performance, it would be worthwhile to compare it with human inspectors' performance.

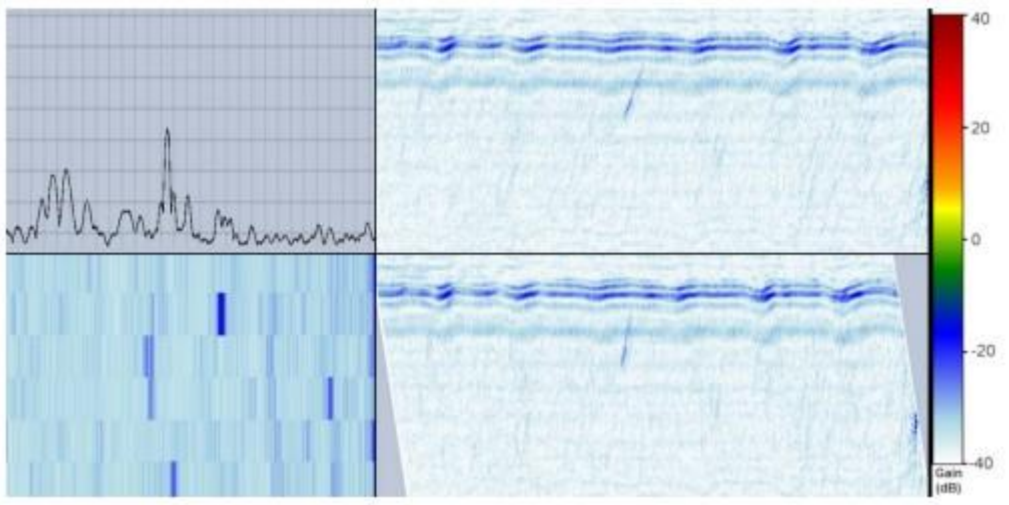

Figure 2.1.3 – Examples of different ultrasonic data representations. Top left: A-scan. Beneath it is the C-scan representation. Right: B-scan (top) and volume corrected B-scan (bottom) (taken from [4]).

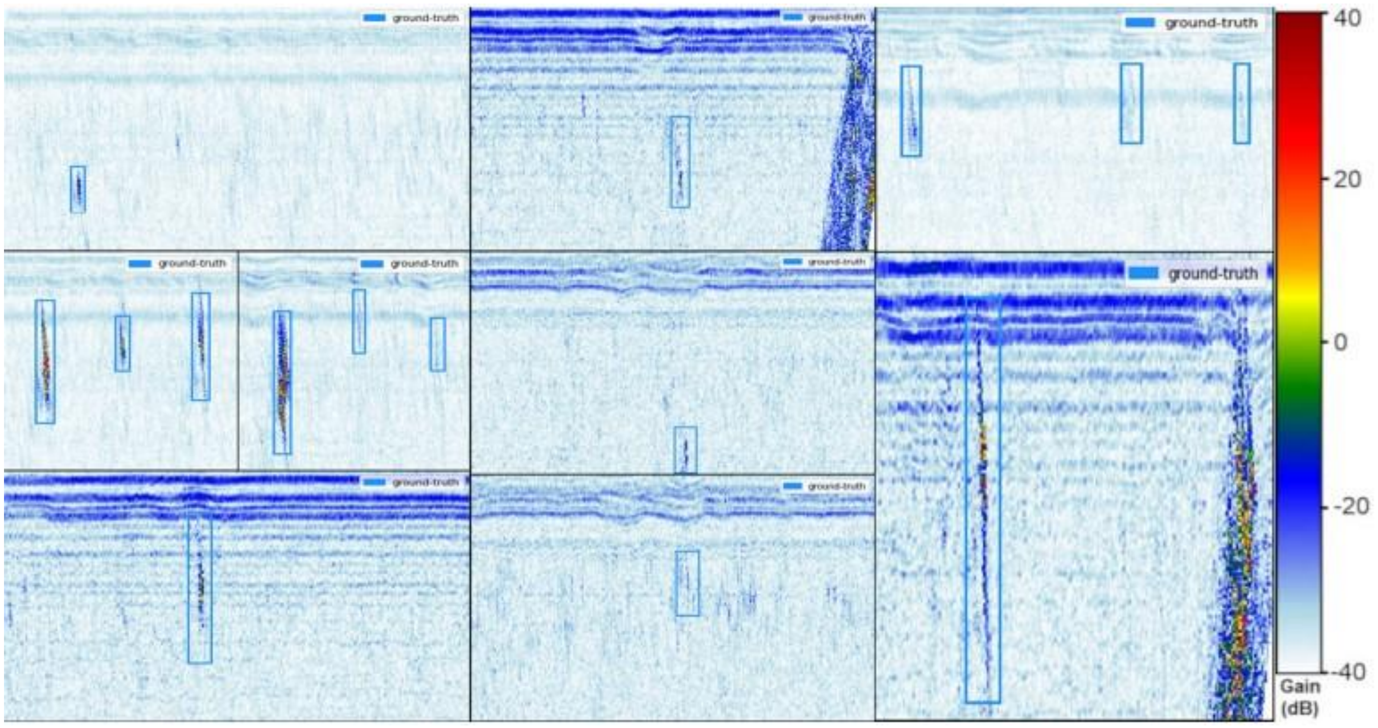

Figure 2.1.4 – Examples of the used VC-B-scans with ground-truth labels (taken from [4]).

In this research[5], a computer vision system for ultrasonic image examination was designed that works with similar accuracy to humans. The researchers started by establishing an ultrasonic inspection image dataset and performing comparative analysis of computer vision methods. The suggested USresNet using deep neural networks surpassed all traditional methods in detection accuracy and model stability. Future work will emphasize the detection of subtle defect patterns from ultrasonic images, and increasing the ultrasonic image database to enhance analysis accuracy.

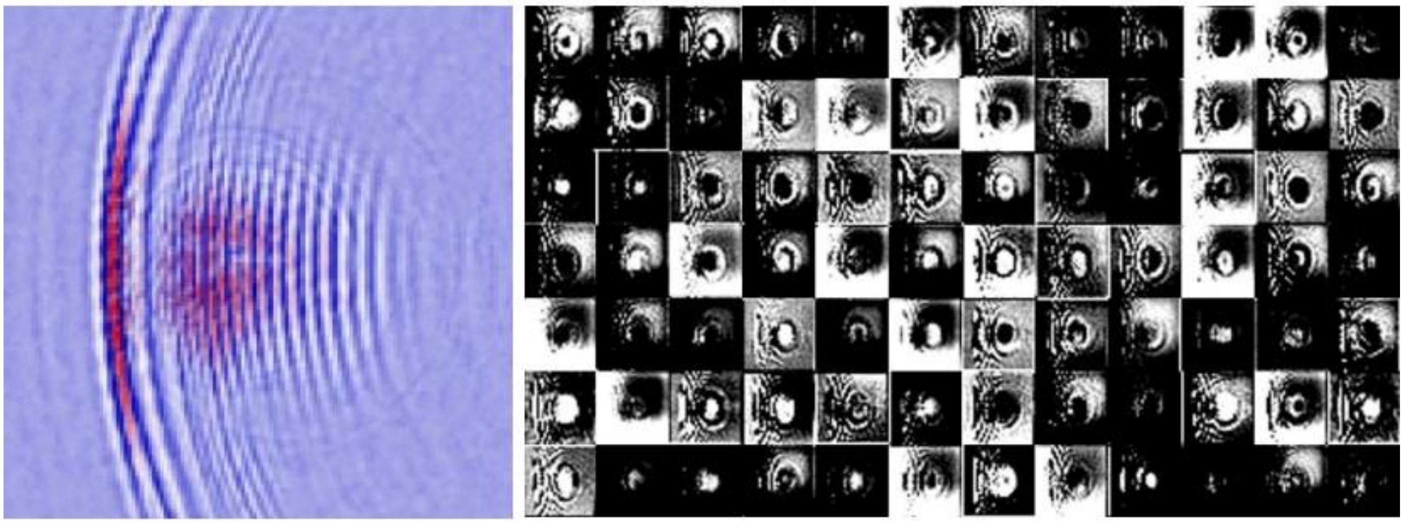

Figure 2.1.5 – Visualization of USresNet B4 layer activation generated by an ultrasonic image with defect (taken from [5])

A novel algorithm based on enhanced YOLOv4[6] has been put forward to identify defects in steel strips. The algorithm applies CBAM and RFB structure to enhance detection accuracy. The model performs better than the latest detection networks, with a 3.87% increase in mAP. But the inclusion and patch detection capability requires enhancement due to a smaller dataset. The researchers intend to extend the sample size to improve detection accuracy.

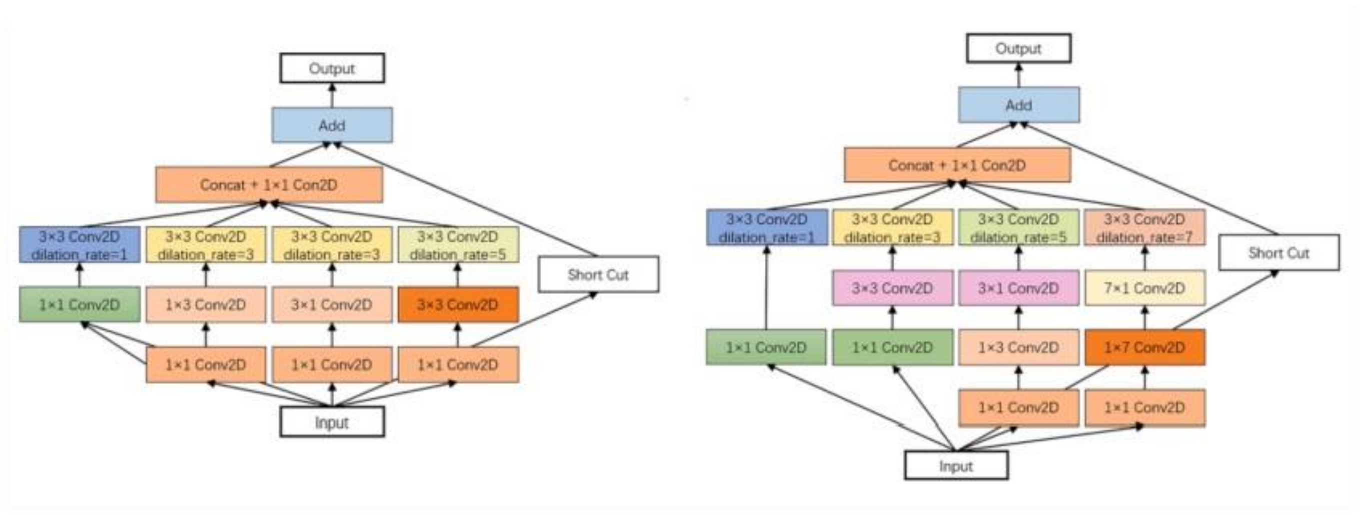

Figure 2.1.6 – Receptive field block(RFB) modifications: (a) RFB for small structures (RFB-S) and (b) our custom RFB. The Customised RFB structure readjusts parameters while deepening the network layer to enhance network generalizability (taken from [6]).

This paper[7] suggests an automatic flaw detection system for steel welded joints based on different methods of feature extraction and classification. The system has two MLP-based classification methodologies, and PCA and WMW test are used for selecting the most significant input features. Experiments indicate that the system effectively determines the defective regions. With the first classifier type, just 20 coefficients out of the initial 2500 A-scan time-samples are needed. With the second approach, very high efficiency in defect signature detection can be attained using merely 5 input features.

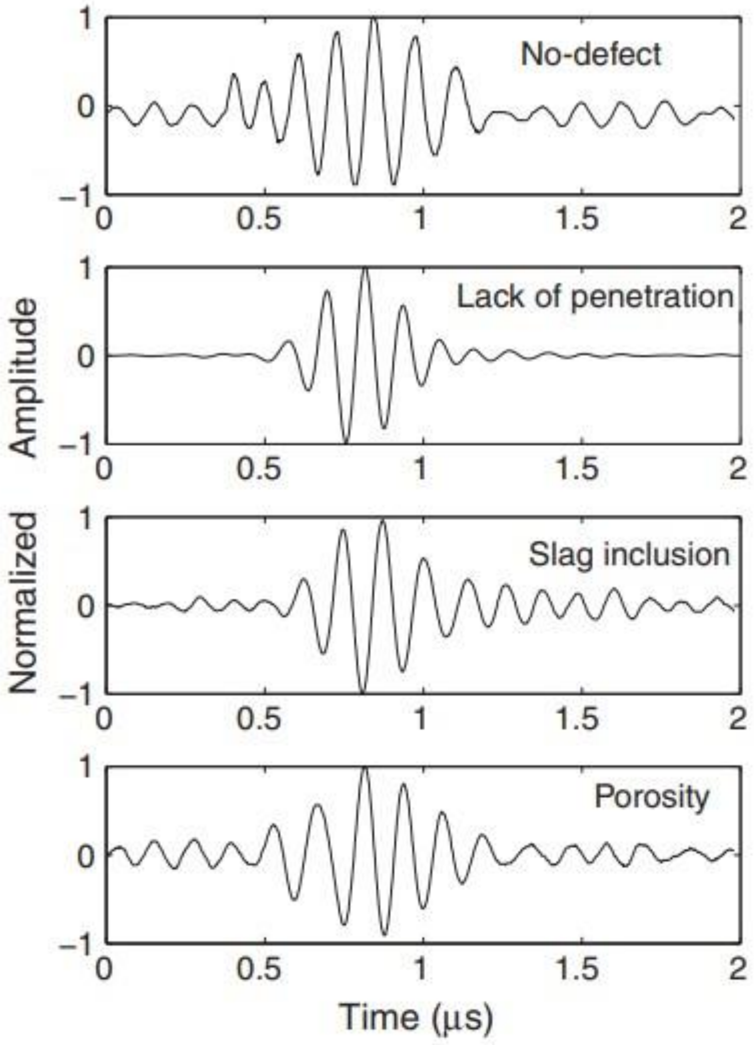

Figure 2.1.7 – Typical A-scan signals from the four classes of interest (taken from [5]).

2.2 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a type of deep learning neural network used for image and video analysis. CNNs function by applying a sequence of convolution and pooling layers to extract features from images and videos. These features, are utilized to classify or detect scenes and objects.

Convolutional Neural Networks (CNNs), are a deep learning algorithm type which is predominantly utilized in the field of pattern recognition.

Following are the most important benefits of convolutional neural networks (CNNs):

1. Excellent at identifying patterns and features in images, videos, and audio signals
2. Automatic feature extraction
3. Highly accurate in image recognition & classification
4. Translation, rotation and scaling invariance robust

5. Reduces computation

6. End-to-end training, no manual feature extraction required

7. Able to manage large volumes of data and provide high accuracy

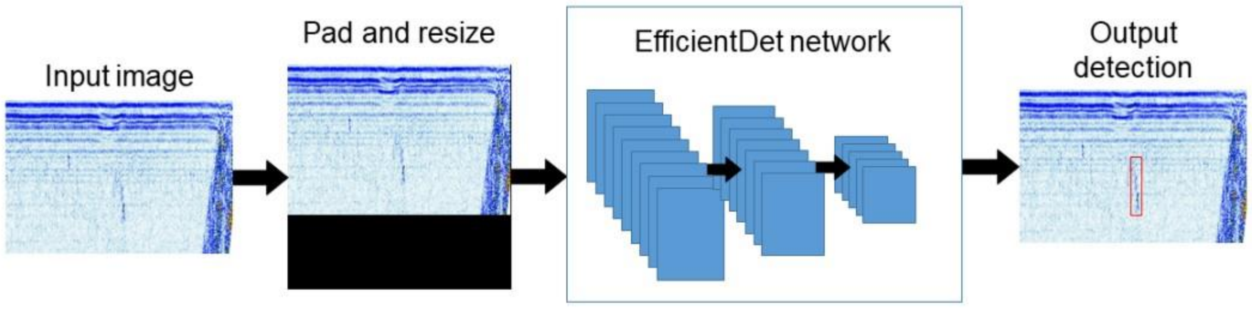

Figure 2.2.1 – Convolutional Network (Taken from [4])

A CNN is a type of network structure for deep learning algorithms and is used particularly for image recognition and image processing-based tasks. Other neural networks are present in deep learning, but for object recognition and identification, CNNs are the network structure of choice.

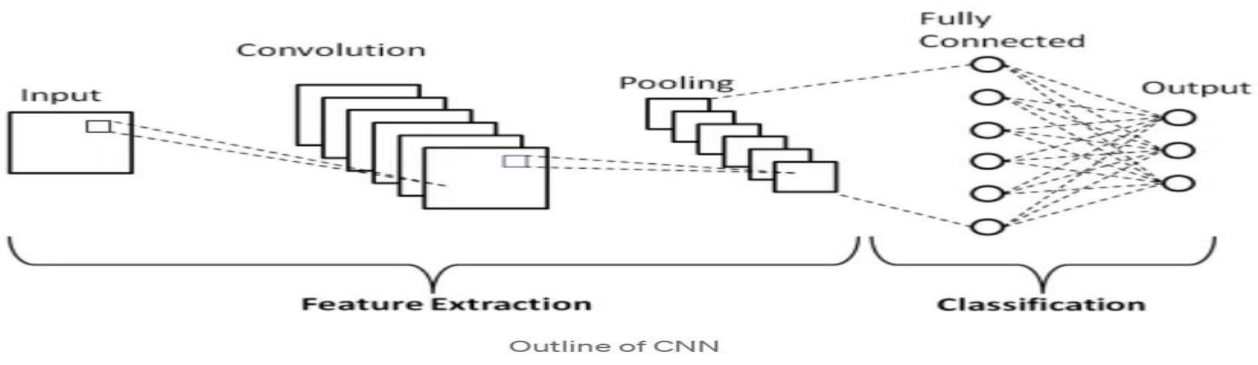

Figure 2.2.2 – Outline of CNN

2.3 Convolutional Layers

The convolutional layer is the initial layer employed to extract different features from input images. In this layer, a mathematical operation known as convolution is conducted between the input image and a filter of a given size (MxM). Through the sliding of the filter over the input image, the dot product is computed between the filter and the corresponding regions of the input image, depending on the dimensions of the filter.

The result of this operation is referred to as the feature map, which contains significant information regarding the image, including its edges and corners. This feature map is then fed into the next layers to learn other features of the input image.

In Convolutional Neural Network (CNN), the convolutional layer passes the outputs to the subsequent layer after performing the convolution operation on the input. Convolutional layers are especially useful since they preserve the spatial relationships among the pixels in the image.

2.4 Pooling Layer

A pooling layer usually comes after a convolutional layer in most instances. The function of this layer is mainly to downsize the convolved feature map, hence reducing computational expenses. It does this by reducing inter-layer connections to its minimum possible value and acting independently on every feature map. Based on the technique applied, there are various kinds of pooling operations, all of which summarize features produced by the convolutional layer.

In max pooling, the most significant element in the feature map is chosen. Average pooling involves finding the mean of the elements in a set portion of the image. Sum pooling finds the sum of the elements in the same set predefined portion. Pooling layer traditionally acts as the interface between convolutional layer and fully connected (FC) layer.

This convolutional neural network (CNN) architecture generalizes the features learned by the convolutional layer so that the network can identify the features autonomously. Consequently, this also minimizes the total computations in the network.

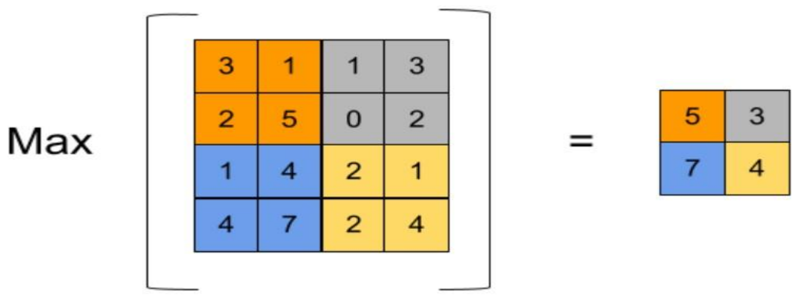

Figure 2.3.1 – MaxPooling

2.5 Fully Connected Layer

The Fully Connected (FC) layer contains weights, biases, and neurons and is employed to connect neurons in two distinct layers of a neural network. These layers are usually located right before the output layer and form the terminal layers of a Convolutional Neural Network (CNN) architecture.

Here, the output image of previous layers is flattened and then passes through the FC layer. One or more than one additional FC layers are utilized here, and several mathematical processes occur here. Here, classification starts. With more than a single fully connected layer, good performance is received compared to utilizing a single layer. Also, these layers of a CNN are utilized to avoid the necessity for human intervention to a great extent.

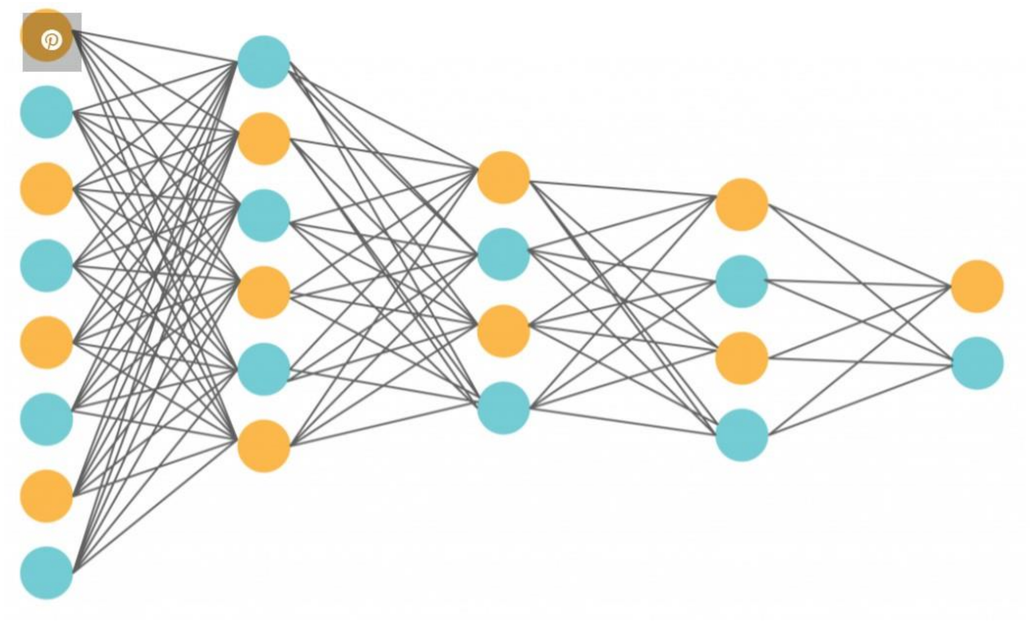

Figure 2.4.1 – Fully Connected Layer

2.6 Dropout

Typically, when all the features are mapped to the FC layer, it may lead to overfitting in the training data. Overfitting happens when a specific model performs so well on the training data that it has a detrimental effect in the model's performance when applied to a new data.

To address this issue, a dropout layer is used where some neurons are dropped from the neural network during training process which results in smaller size of the model. When passing through a dropout of 0.3, 30% of the nodes are randomly dropped out from the neural network.

Dropout works by enhancing the performance of a machine learning model by avoiding overfitting because it simplifies the network and prevents it. It removes neurons from the neural networks while they are being trained.

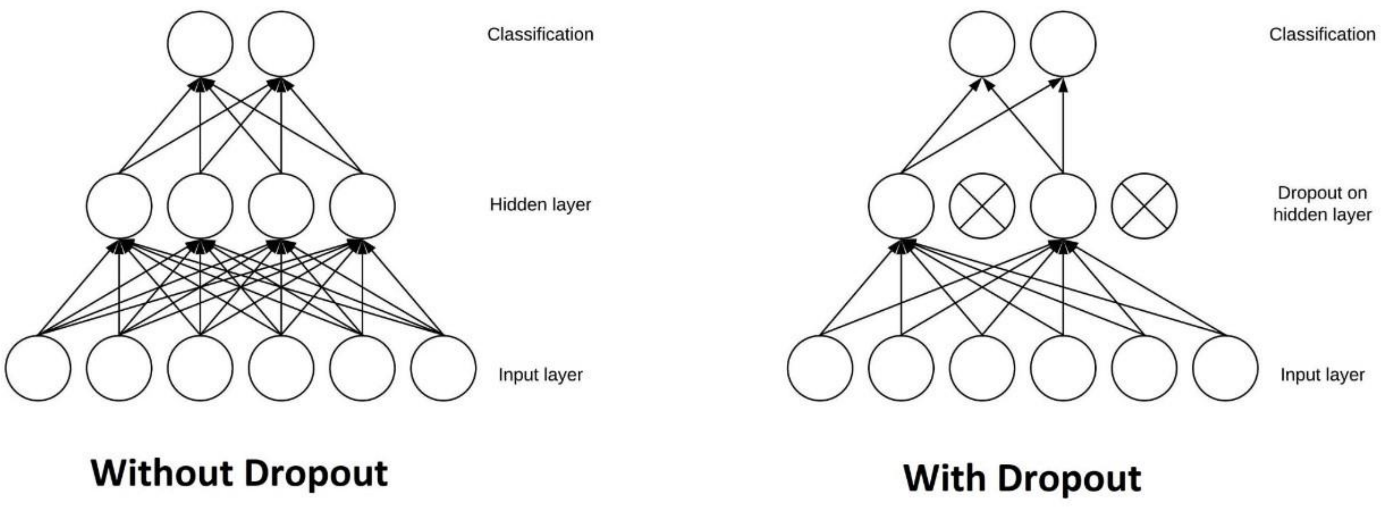

Figure 2.5.1 – Dropout Layer

2.7 Activation Functions

A parameter of the utmost significance of a CNN model is the activation function. Activation functions are employed in learning and the approximation of high-level relationships among the variables in the network. Simply put, they determine the information in the model that gets activated during the forward pass and what does not.

These operations bring non-linearity to the network. Some of the most commonly employed activation functions are ReLU, Softmax, tanh, and Sigmoid. Each of these functions is used for certain purposes. In a CNN model, for binary classification, the Sigmoid and Softmax functions are utilized, but for multi-class classification, Softmax is mostly used.

In short, activation functions in a CNN model decide whether a neuron must be activated. They evaluate the significance of the input data in making predictions through mathematical operations.

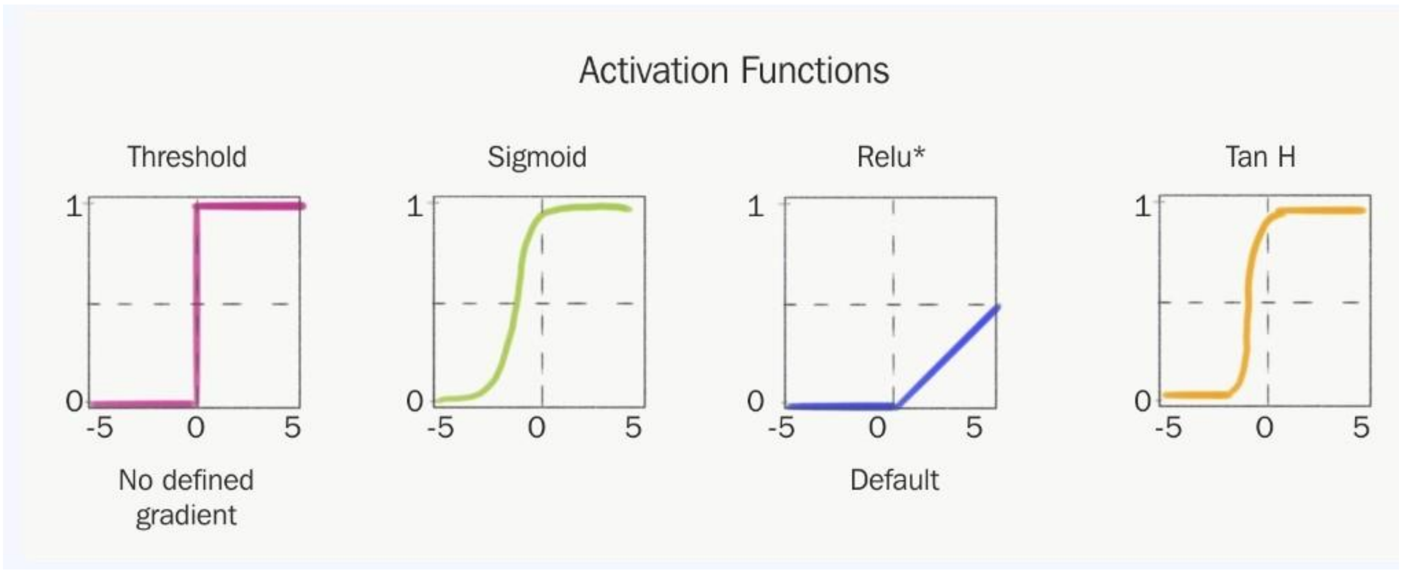

Figure 2.5.1 – Activation Functions

Chapter 3

Dataset

3.1 Estimation of NDT performance and probability of detection(POD)

NDE is worthiest when applied to area, whose expected reliability is extremely high. Therefore, it is challenging to measure the performance of an NDE system and its reliability, specifically. It calls for high number of evaluation results on pertinent targets and hence, high number of test samples with representative defects. Supplying these imperfect test samples is expensive and hence various methods have developed to make the best out of the rest blocks available. Presently, the conventional method of quantifying NDE performance is to establish a probability of detection (POD) curve and, specifically, the minimum crack that can be detected at level of adequate confidence, generally 90% POD at 95% confidence (a90/95). Experimentally, the POD curve is established with test block trials and a collection of standardized statistical software.

3.2 NDT Data

Test specimen inspected for data-acquisition was a butt-weld in austenitic 316L stainless steel pipe. Three thermal fatigue cracks with depths 1.6, 4.0 and 8.6 mm were introduced in the inner diameter of the pipe close to the weld root by Trueflaw ltd. and ultrasonically scanned. Austenitic weld has been selected as a test specimen because it is prevalent in industry. Besides, austenitic weld has higher inspection complexity due to noise generated by the anisotropy of the weld structure. Inspection technique employed for data acquisition was Transmission Receive shear (TRS) phased array, one of the popular techniques employed in inspection of austenitic and dissimilar metal welds. The scan was performed using Zetec Dynaray 64/64PR-Lit flaw detector connected to a PC. Probes utilized were a Imasonic 1.5MHz 1.5M5x3E17.5-9 matrix probes with center frequency at 1.8MHz, element size 3.35 x 2.85 mm and element configuration as 5 x 3 elements. Wedge utilized was ADUX577A utilized to create a shear wave effectively. Single linear scan without any skew angles was employed. The ultrasonic wave was directed to the inner surface of the pipe and the probe was oriented so that the beam would be directed directly to the produced cracks. Coupling was done using a feed water system and the pipe was rotated under the probe to ensure constant and uniform coupling between the pipe and the probe. Probe position was monitored carefully in the scan direction by Zetec pipe scanner with 0.21 mm scan resolution. The specimen and the inspection method are explained more in detail in Koskinen et al. (2018). The specimen and the scanner are presented in Figure 1.

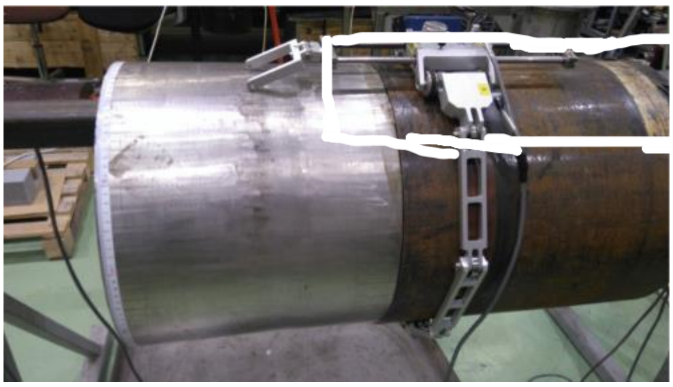

Figure 1: Scan set-up with Zetec pipe scanner, extension fixed to the right side for scanner mounting.

3.3 Training and Validating Data

An ultrasonic dataset comprising manufacturing flaw data was utilized for developing and validating a deep learning model. The dataset consists of 796 training files and 8 validation files, each identified by a unique UUID. Each file in the dataset is associated with several components: a .bins file containing raw data in uint16 format with dimensions of 256 x 256 x 100, a .meta file documenting the raw data format, a .jsons file containing metadata such as flaw locations, source flaw size, and "equivalent size," and a .labels file providing tab-separated data indicating flaw existence (0 or 1) and equivalent flaw size. This comprehensive dataset structure facilitated the training and evaluation of the deep learning model for accurate detection and characterization of manufacturing flaws using ultrasonic imaging data.

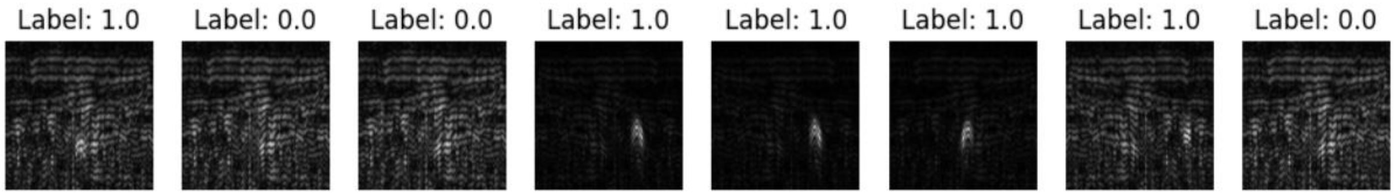

Figure 2: Ultrasonic Image with Labels

Chapter 4

Model Implementation

4.1 ResNet50

The ResNet50 model is a convolutional neural network (CNN) architecture widely known for its effectiveness and efficiency in image classification tasks. ResNet50 refers to the fact that it has 50 layers, making it a deeper model compared to ResNet18 and ResNet34 but lighter than ResNet101 and ResNet152. This specific architecture employs residual learning, where shortcut connections (or "skip connections") allow the model to bypass certain layers. This design addresses the problem of vanishing gradients and allows the model to train effectively even with a large number of layers.

Compared to other ResNet variants, ResNet50 strikes a balance between computational efficiency and model complexity. ResNet18 and ResNet34 are relatively shallow and may struggle with complex datasets, while ResNet101 and ResNet152 are deeper and can capture more intricate patterns, but they require more computational power and may lead to overfitting on smaller datasets. ResNet50 is a good compromise, capturing sufficient detail while being computationally feasible for most applications, making it a popular choice in many image classification tasks.

The plot shows the training and validation accuracy over 50 epochs with the ResNet50 model. Training accuracy improves quickly and reaches nearly 100%, indicating that the model learns the training data well. However, the validation accuracy remains flat at around 50%, which strongly suggests overfitting. This gap between training and validation performance indicates that the model may have memorized the training data without generalizing to unseen data. Possible reasons for this could include insufficient regularization, lack of data augmentation, or an inadequate validation dataset.

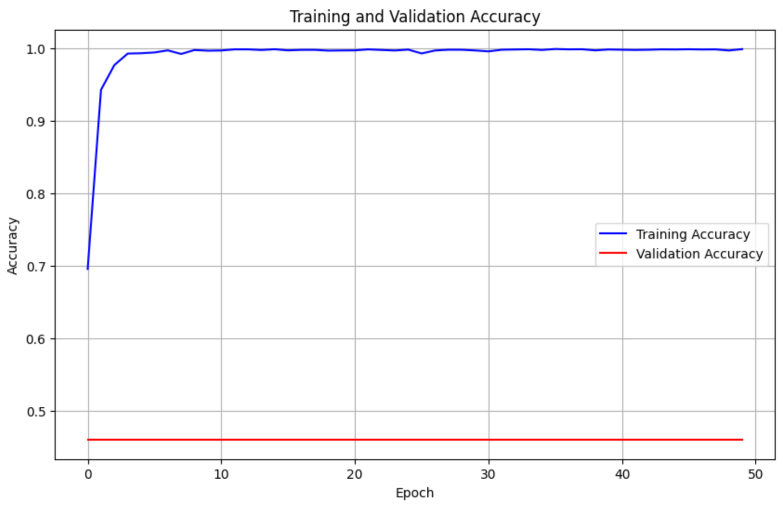

Figure 4.1.1 – ResNet50 - Epoch vs Accuracy

4.2 VGG-19

VGG19 consists of 19 layers with learnable weights, including 16 convolutional layers and three fully connected layers. The architecture uses small, fixed-size convolutional filters (3x3) and follows a consistent pattern of stacking convolutional layers followed by max-pooling layers. This design emphasizes depth and simplicity, allowing VGG19 to capture intricate patterns and complex features in images, making it a strong choice for a variety of image classification tasks.

According to the training and validation accuracy graph, VGG19 appears to yield better results than ResNet50 in this specific case. The training accuracy of VGG19 remains consistently high, with minimal fluctuations, suggesting that the model is effectively learning the patterns in the training data. Although the validation accuracy varies more, it still maintains a high level overall, indicating that the model generalizes well to unseen data. This stability could be attributed to VGG19’s structured convolutional layers, which allow it to capture features consistently across different samples.

Several factors might explain why VGG19 outperforms ResNet50 here. First, the dataset characteristics could favor a simpler architecture like VGG19 over the more complex residual connections in ResNet50. The specific patterns and structures in the images might be better suited to VGG19's straightforward, depth-based approach rather than ResNet50’s skip connections, which excel in cases where the depth of the network might lead to diminishing gradients or the need for finer detail preservation. Additionally, VGG19's dense, fully connected layers at the end of the network could be particularly effective in the final stages of classification, enhancing its ability to distinguish between classes in the dataset. The variations in validation accuracy also suggest that while VGG19 may be slightly overfitting at times, its overall performance is still robust, potentially due to effective regularization techniques like dropout and batch normalization.

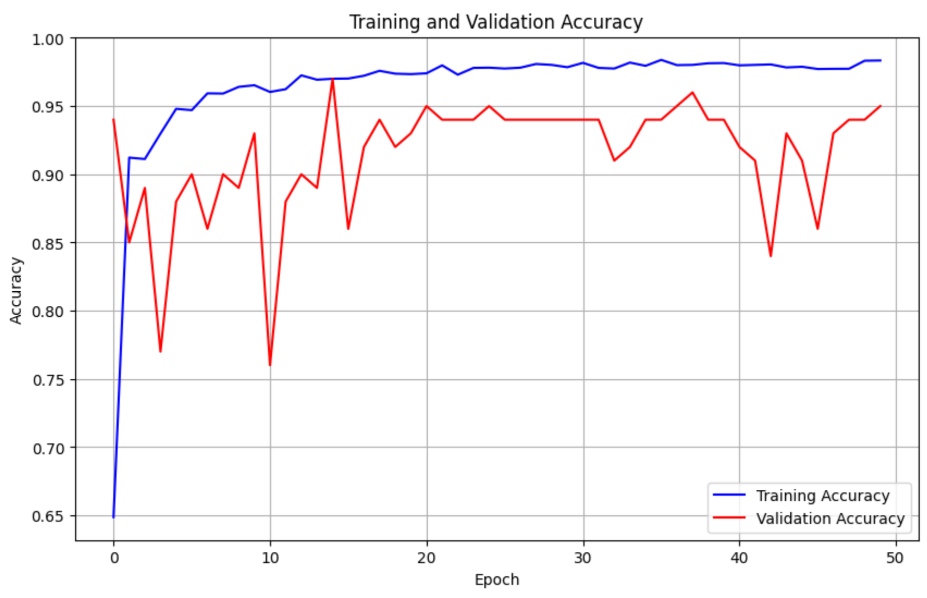

Figure 4.2.1 – VGG19 Model – Epoch vs Accuracy

4.3 Custom CNN Architecture

The custom convolutional neural network architecture used in this project was designed to effectively classify ultrasonic images data for defect detection tasks. It begins with an input layer that processes 256 x 256 grayscale ultrasonic images. The model include a combination of traditional 2D convolutional layers (Conv2D) and depth-wise separable convolutional layers (SeparableConv2D), which are known for its computational efficiency and ability to extract fine-grained features. These convolutional operations are alternated with BatchNormalization layers to stabilize and accelerate training, and ReLU activation functions to introduce non-linearity into the network. The architecture utilizes a series of filters with decreasing channel sizes (128, 94, 64, 32), enabling the model to capture high and low-level features throughout its depth. The MaxPooling2D layers with a (2, 2) stride are applied to progressively reduce the spatial dimensions of the feature maps, preserving the most important information while minimizing computational complexity.

Following the feature-extraction phase, the model uses GlobalAveragePooling2D, which condenses each feature map to a single value, effectively summarizing spatial data and reducing overfitting. The pooled features are then passed through fully connected (Dense) layers with ReLU activation and Dropout regularization to enhance generalization. The final layer utilizes a sigmoid activation function to output a probability score for binary classification, indicating the presence or absence of a defect in the ultrasconic image. The model is optimized using the Adam optimizer with a low learning rate of 0.0001 and clipnorm to prevent exploding gradients. Overall, this custom CNN architecture is designed in such a way that to handle the complexity of ultrasonic data, ensuring high accuracy and strong generalization capability as evidenced by its performance on both training and validation datasets.

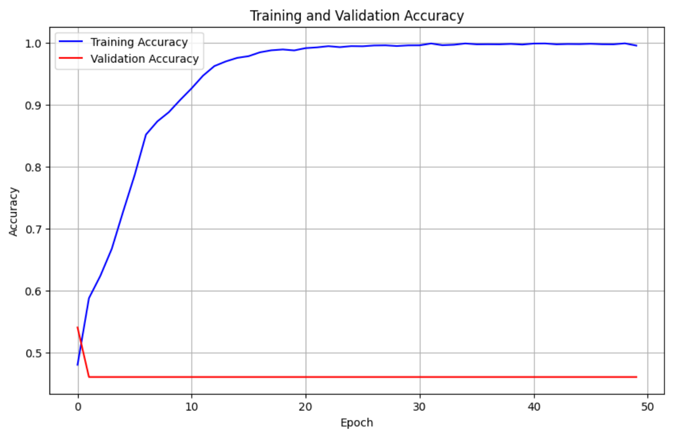

Figure 4.3.1 – Initial Model(Epoch Vs Accuracy)

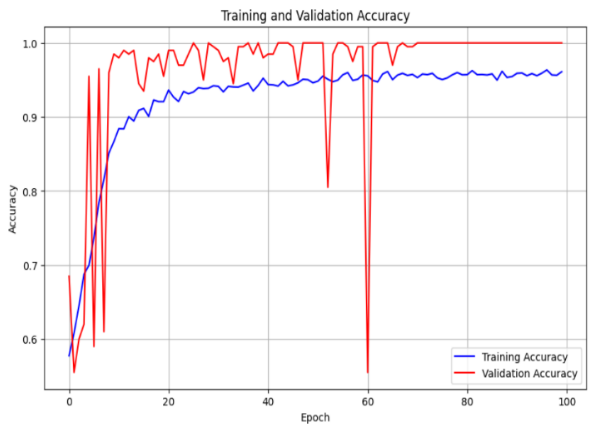

Figure 4.3.2 – Updated Model

With steps per epoch =500

For a Conv2D / Pooling layer:

$$\text{Output Height (H)} = \left\lfloor \frac{H_{in}+2P-k}{s} + 1 \right\rfloor \quad \text{.......................... (Eq. 4.3.1)}$$

$$\text{Output Width (W)} = \left\lfloor \frac{W_{in}+2P-k}{s} + 1 \right\rfloor \quad \text{.......................... (Eq. 4.3.2)}$$

Where:

H_{in}, W_{in}: Input Height & Width

P: Padding

k: Kernel Size

s: Stride

Layer (type)	Output Shape	Parameters
input_layer (InputLayer)	(None, 256, 256, 1)	0
max_pooling2d (MaxPooling2D)	(None, 36, 256, 1)	0
conv2d (Conv2D)	(None, 36, 256, 128)	256
batch_normalization (BatchNormalization)	(None, 36, 256, 128)	512

activation (Activation)	(None, 36, 256, 128)	0
conv2d_1 (Conv2D)	(None, 34, 254, 128)	147,584
batch_normalization_1 (BatchNormalization)	(None, 34, 254, 128)	512
activation_1 (Activation)	(None, 34, 254, 128)	0
separable_conv2d (SeparableConv2D)	(None, 34, 254, 94)	13,278
batch_normalization_2 (BatchNormalization)	(None, 34, 254, 94)	376
activation_2 (Activation)	(None, 34, 254, 94)	0
max_pooling2d_1 (MaxPooling2D)	(None, 17, 127, 94)	0
separable_conv2d_1 (SeparableConv2D)	(None, 17, 127, 94)	9,776
batch_normalization_3 (BatchNormalization)	(None, 17, 127, 94)	376
activation_3 (Activation)	(None, 17, 127, 94)	0
separable_conv2d_2 (SeparableConv2D)	(None, 17, 127, 64)	6,926
batch_normalization_4 (BatchNormalization)	(None, 17, 127, 64)	256
activation_4 (Activation)	(None, 17, 127, 64)	0
max_pooling2d_2 (MaxPooling2D)	(None, 8, 63, 64)	0
separable_conv2d_3 (SeparableConv2D)	(None, 8, 63, 64)	4,736
batch_normalization_5 (BatchNormalization)	(None, 8, 63, 64)	256
activation_5 (Activation)	(None, 8, 63, 64)	0
separable_conv2d_4 (SeparableConv2D)	(None, 8, 63, 32)	2,656
batch_normalization_6 (BatchNormalization)	(None, 8, 63, 32)	128
activation_6 (Activation)	(None, 8, 63, 32)	0
separable_conv2d_5 (SeparableConv2D)	(None, 8, 63, 32)	1,344

batch_normalization_7 (BatchNormalization)	(None, 8, 63, 32)	128
activation_7 (Activation)	(None, 8, 63, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dense (Dense)	(None, 16)	528
RNN (Dense)	(None, 8)	136
dropout (Dropout)	(None, 8)	0
dense_1 (Dense)	(None, 1)	9

Table 4.3.1 – Custom CNN Architecture

The performance of the two models can be measured using the training and validation accuracy plots. On the Figure 4.3.1, the training accuracy rises steeply and is almost 100%, while the validation accuracy is flat at 46–50% during the whole training period. This wide difference between training and validation accuracy strongly shows overfitting, in which the model has memorized the training set but is not able to generalize to new data. This might be due to causes such as too complex of a model, failure to utilize regularization methods, or varying training and validation data distributions. In contrast, the performance of the Figure 4.3.2 is extremely well-balanced. The training accuracy converges to 93–95%, and the validation accuracy also increases by a great extent, reaching nearly 100% for numerous epochs. While there are occasional spikes and dips in the validation curve, the general trend indicates robust generalization. These variations could be caused by batch-level noise or sensitivity of the validation process, but the model still performs well overall. Compared to the first model, the second model shows improved learning and adaptability and is therefore more suitable for deployment or further development.

The dataset for this model contains 796 training files and just 8 validation files, and this will have a direct effect on the nature of the validation accuracy plot. With only so many validation files, each image individually contributes 12.5% to the total validation accuracy and thus the metric becomes very sensitive. This sensitivity is evidenced in the patterns seen in the plots—flatlines in the Figure 4.3.1 show that the model is consistently misclassifying the same set of validation samples, so no improvement in accuracy is visible. In contrast, the Figure 4.3.2 displays random spikes and dips, which happen because even a single image being correctly or incorrectly classified can result in an abrupt 12.5% change in accuracy. These phenomena indicate the difficulty of measuring model performance on an imbalanced dataset in which the validation set is too small to offer a stable or representative measure of generalization.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{(Eq. 4.3.3)}$$

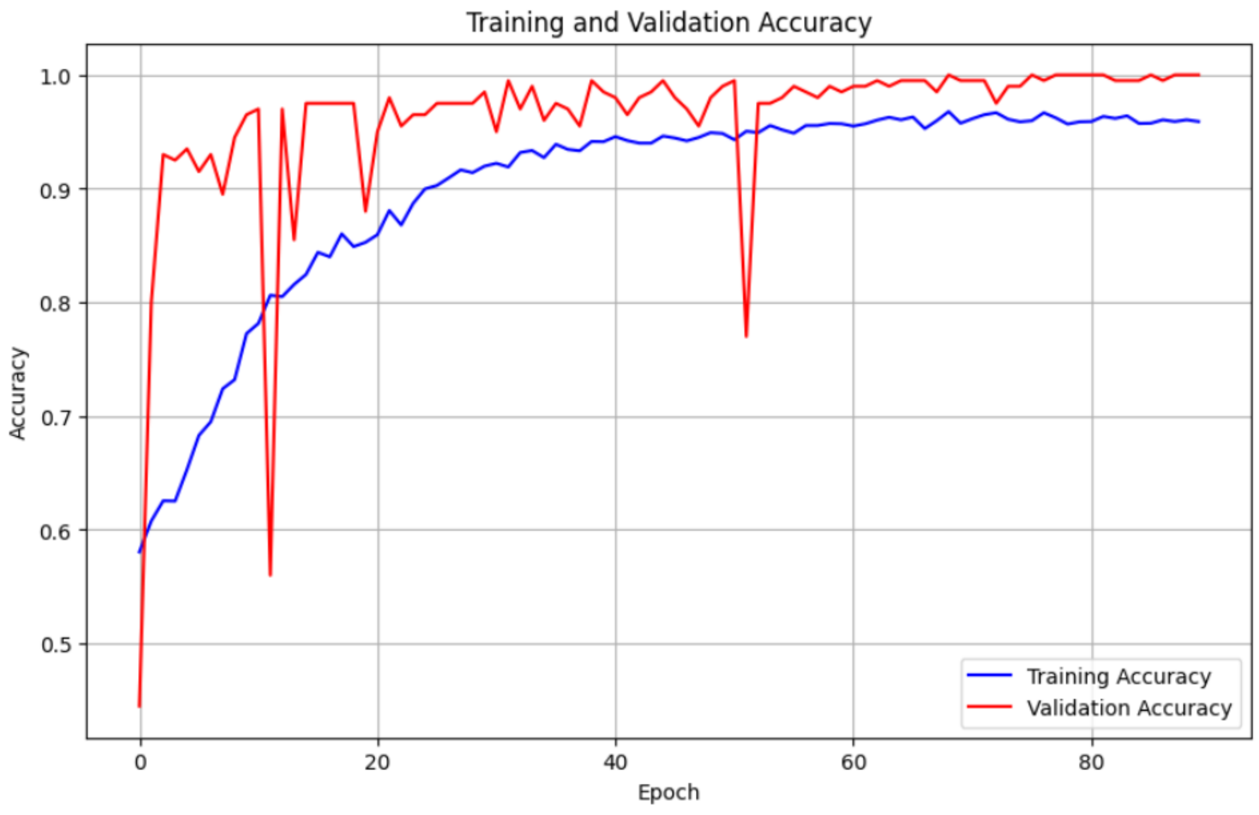

Figure 4.3.3 – Updated Model – Epoch(90 with steps per epoch = 400) vs Accuracy

4.4 Precision

Precision chart (Figure 4.4.1) for the current model shows strong differences between validation and training performance. The precision while training goes steadily up and settles close to 1.0, illustrating how well the model is classifying positive instances as it learns. The precision of validation goes sharply up and down during training and is totally different from precision on the train data. This volatility comes from the extremely tiny validation set of just 8 samples, where every picture has a very significant impact on the metric—getting one misclassified can lower precision by 12.5%. These volatile fluctuations show how the performance of the model over such a minuscule set can seem random, frequently featuring flatlines with the same misclassifications happening or sudden jumps when those get fixed.

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{.......................... (Eq. 4.4.1)}$$

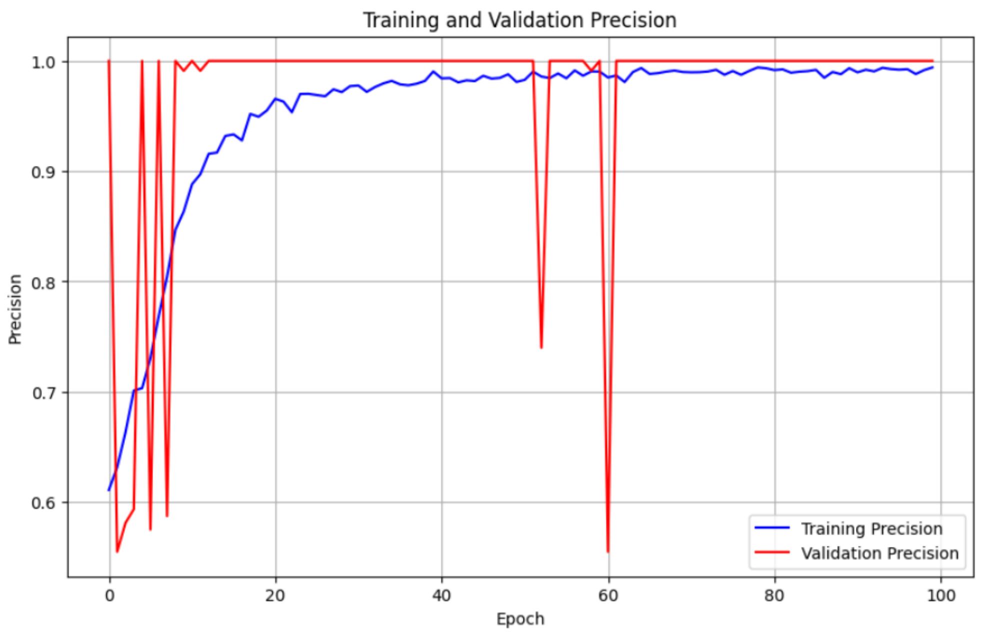

Figure 4.4.1 – Epoch vs Precision

4.5 Recall

The recall graph (Figure 4.5.1) illustrates the sensitivity of the model in recognizing positive instances. The training recall settles at 0.93, reflecting constant performance in true positive identification in the training set. The validation recall, however, varies erratically in the initial epochs before ultimately stabilizing at around 1.0. These early fluctuations are mostly because of the small size of the validation set—only 8 samples—where a single error in classification can result in a large decrease in recall, and fixing it can immediately increase the metric. In spite of these early fluctuations, the last part of the graph indicates good and stable generalization regarding recall, where the model consistently classifies all or almost all positive cases.

$$\text{Recall} = \frac{TP}{TP + FN} \quad \text{.......................... (Eq. 4.5.1)}$$

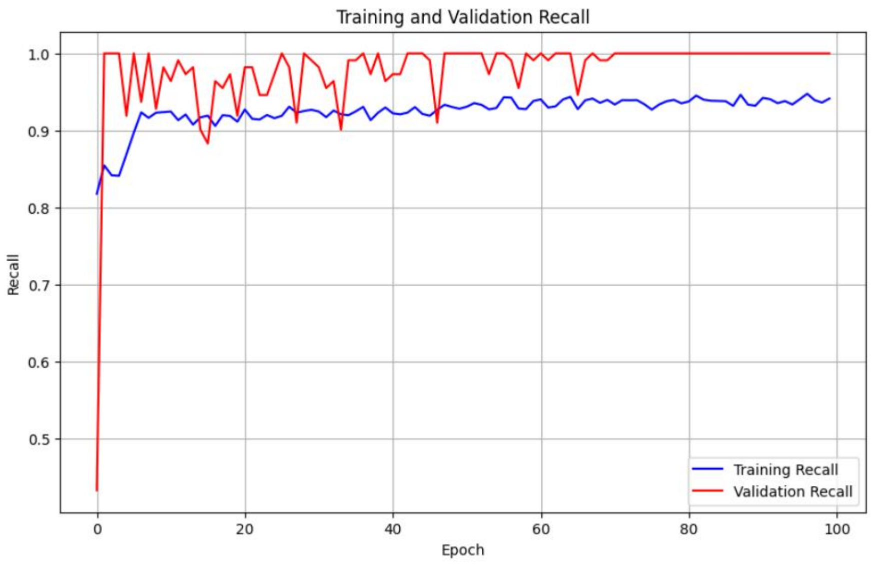

Figure 4.5.1 – Epoch vs Recall

4.6 AUC

The AUC graph (Figure 4.6.1) shows how well the model separates classes over time. Training AUC stabilizes at about 0.93, meaning it is consistently ranking positive examples above negative ones. Validation AUC fluctuates early on, from sudden dips to perfect discrimination (AUC = 1.0). These spasmodic fluctuations are once again due to the extremely tiny size of the validation set—8 samples only, where one solitary misclass is enough to adversely affect the score. Nonetheless, as training evolves, so too does the AUC in validation stabilize at or around 1.0, indicating perfect generalization from the perspective of separability of the classes. As a whole, the model performs steadily and possesses high discriminant ability over training and validation set alike.

$$\text{TPR} = \text{Recall} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(Eq. 4.6.1)}$$

$$\text{FPR} = \frac{FP}{FP+TN} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(Eq. 4.6.2)}$$

$$\text{AUC} = \int_0^1 T\,PR(FPR)d(FPR) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(Eq. 4.6.3)}$$

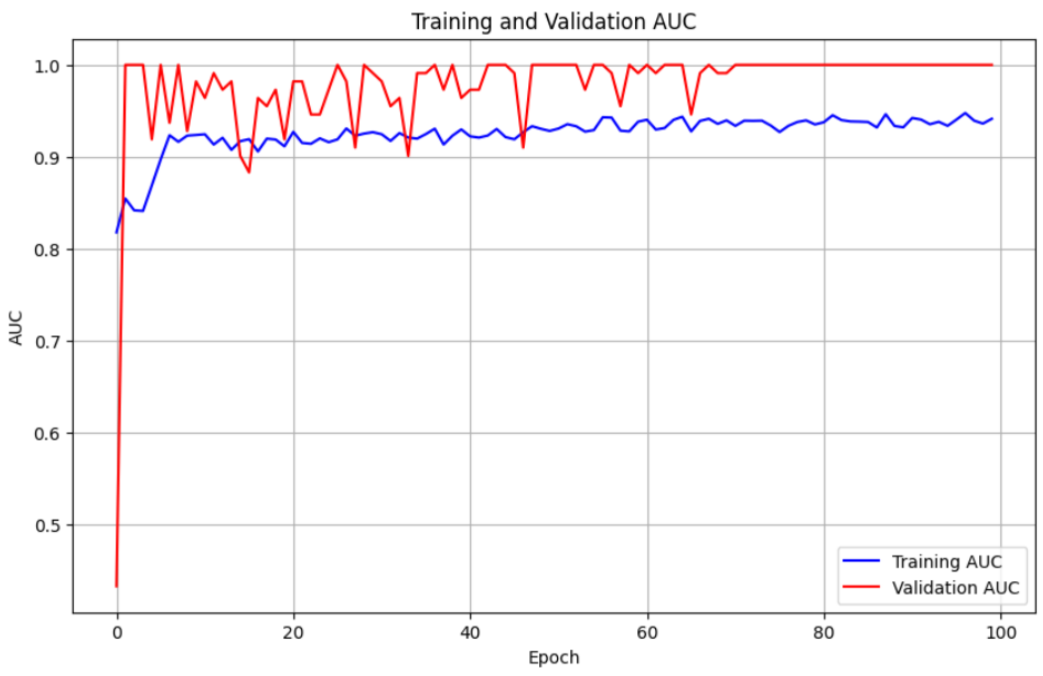

Figure 4.6.1 – Epoch vs AUC

4.7 Results

Training and validation performance of the model were measured in terms of Accuracy, Recall, and AUC measures across 100 epochs, where each epoch consisted of 500 steps (steps_per_epoch = 500). The model reflected robust and uniform performance in all three measures. Training Accuracy quickly settled down around 93%, closely tracking with Training Recall and AUC as well, which also stayed in values close to 0.93. The validation metrics indicated significant oscillations in the early epochs, mainly because of the limited size of the validation set of just 8 samples—where a single misclassification can have a significant effect on metric values. In spite of this initial volatility, Validation Accuracy, Recall, and AUC all improved and stabilized progressively, with all three metrics converging to values near 1.0. This means that the model did not just train well but generalized well to test data, possessing high reliability and robustness to many performance measures.

Chapter 5

Conclusion

The development and evaluation of a deep learning model for defect detection in manufacturing based on ultrasonic images have shown great potential. With the extensive dataset of 796 training files and 8 validation files, including raw data, metadata, and labels, the model was effectively trained and validated. The model performed well consistently throughout 100 training epochs, each consisting of 500 steps, with high performance on Accuracy, Recall, and AUC measures. Although early instability was seen in the validation performance due to the small size of the validation sample, the measures settled down over time to converge to values close to 1.0—showing high generalization ability. These results verify the capability of the model to identify defects in manufacturing accurately using ultrasonic imaging and indicate its readiness for real-world implementation in quality assurance processes.

Chapter 6

Future Work

Though the existing deep learning model is found to exhibit great accuracy and stability in the identification of manufacturing faults from ultrasonic images, there are some areas for future research. One such important direction is increasing the validation dataset size with a diverse and representative sample population, which will improve the generalizability and stability of the model under testing. Also, the inclusion of real-time deployment functionality and edge-based inference can enhance the practicality of the model in on-site industrial settings. Additional experimentation with newer architectures like attention mechanisms, transformers, or self-supervised learning techniques might also enhance performance. Additionally, using domain-specific data augmentation methods and investigating multi-class classification for various types of defects might yield further insights and enhance the model's real-world applicability in various manufacturing contexts.

Chapter 7

References

[1] **Anbesh Jamwal, Rajeev Agarwal, Monica Sharma (2022)**: Deep learning for manufacturing sustainability: Models, applications in Industry 4.0 and implications, Vol. 2, Issue 2

[2] **Ashraf Abou Tabl, Abedalrhmaan Alkhateeb, W. H. Elmaragraphy** : Deep Learning Method based on Big Data for Defects Detection in Manufacturing Systems Industry 4.0, Vol. 2, 1-14

[3] **Luka Posilovic, Duje Medak, Fran Milkovic, Marko Subasic, Marko Budimir, Sven Loncaric**: Deep learning-based anomaly detection from ultrasonic images, Vol. 124

[4] **Duje Medak, Luka Posilovic, Marko Subasic, Marko Budimir, Sven Loncaric**: Automated Defect Detection From Ultrasonic images Using Deep Learning Vol.68, Issue:10

[5] **Jiaxing Ye, Shunya Ito, Nobuyuki Toyama**: Computerized Ultrasonic Imaging Inspection: From shallow to deep learning.

[6] **Virkkunen, I., Haapalainen, J., Papula, S., Sarikka, T., Kotamies, J., and Hanninen, H. (2017):** Effect of feedback and variation on inspection reliability. In 7th European-American Workshop on Reliability of NDE. German Society for Non-Destructive Testing.