Master of Science

O-RING DEFECT INSPECTION FRAMEWORK BASED ON DEEP LEARNING ALGORITHMS

The Graduate School of the University of Ulsan

Department of IT Convergence

MUSTAFAEV BEKHZOD GOFUROVICH

O-RING DEFECT INSPECTION FRAMEWORK BASED ON DEEP LEARNING ALGORITHMS

Supervisor: Professor Ja-Rok, KOO

A Thesis

Submitted to

The Graduate School of the University of Ulsan

In partial Fulfillment of the Requirements for the Degree of

Master of Science

By

MUSTAFAEV BEKHZOD GOFUROVICH

Department of IT Convergence

University of Ulsan, Korea

NOVEMBER 2020

O-RING DEFECT INSPECTION FRAMEWORK BASED ON DEEP LEARNING ALGORITHMS

This certifies that this Thesis of Mustafaev Bekhzod Gofurovich is approved by:

Committee Chair: Professor Jae-Hak, Bae:

_____

Committee Member: Professor Seokhoon, Yoon:

_____

Committee Member: Professor Ja-Rok, Koo:

_____

Department of IT Convergence

University of Ulsan

NOVEMBER 2020

ABSTRACT

O-ring defect inspection is a challenging task because of their variable sizes as well as defect types. Traditional quality inspection systems use image processing to extract hand-crafted features and machine learning models to classify defect types. However, they are very sensitive to different sizes of O-ring objects and various defect types. Moreover, software maintenance of traditional systems is hard and requires much effort and time. To solve those challenges, we proposed a fully deep learning (DL) – based O-ring defect inspection framework. Proposed system, firstly, uses DL object localization model to find the location of O-ring objects in an image, then inspects it via using DL classification model. We used pre-trained Convolutional Neural Networks (CNN) for both object localization and defect classification and trained using transfer learning technique. The proposed system is evaluated by using our custom O-ring dataset. Object localization was achieved 98.5 % accuracy on test data. 97 % accuracy was achieved on a classification task. From the experimental results, we see that proposed DL-based system is capable of the inspection of O-ring objects accurately and robust for the various O-ring objects as well as defect types.

ACKNOWLEDGEMENT

I would like to express deep gratitude to my supervisor Prof. JA-ROK, KOO, and my academic advisor Prof. T.H.RHO for their guidance, encouragement and gracious support throughout the course of my work, for their expertise in the field that motivated me to work in this area and for their faith in me at every stage of this research.

I would like to thank all my friends who always supported me at tough times and help to make my Master's journey wonderful. Moreover, also all the staff and professors of Department of IT convergence for giving their guidance on the right time and taught us well. Finally, I would like to thank my parents and brothers for their continued support whenever I had a hard time.

Republic of Korea, Ulsan, NOVEMBER 2020

MUSTAFAEV BEKHZOD GOFUROVICH

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter 1: INTRODUCTION

1.1 General Context

O-rings are the small rubber seals that are very widely used in the industry, starting from an individual tiny seal for repairs or maintenance to top-quality assured applications in aerospace, automotive or general engineering. A quality of Sealing is a significant measure for estimating the equipment quality and directly affects the accuracy and stability of the device. Therefore, vouching for the quality of the O-rings is significantly important [1].

Automatic defect inspection using image processing and traditional machine learning is being widely adopted in many industries. To improve the inspection quality many researchers used several useful methods over the past years. Martinez et al. concentrated on the optimization of the visual examination of piston surface defects, the light source. [2] Roby et al. [3] proposed an inspection method of extremely insightful chrome-coated rings used in machine vision textile equipment. A suitable real-time defect detecton and localization algorithm was investigated by Li et al. [4] on surfaces of circular object sequences. Gaoliang Peng, et al. also had a good approach that they proposed both hardware and software inspection method for O-rings using traditional image processing and machine learning algorithm[5]

For the inspection of different industrial goods, Deep Learning-based methods have shown very good results in recent years. In 2012, Krizhevsky et al. made the breakthrough with AlexNet in object recognition by winning the ImageNet Broad Scale Visual Recognition (ILSVRC) rivalry with CNNN for the first time.[6] Masci et al. introduced Max-Pooling CNN model approach for supervised steel defect classification [7]. Later, in a different approach, Fatima A. Saiz and Ismail Serrano et al. Combining traditional Machine Learning techniques with CNN has yielded good results and had focused on increasing robustness and accuracy in detecting steel surface defects [8]. Recently, Chao-Ching Ho*, and Eugene Su at. al has been introduced the Deep Learning-based approach of Rubber Gasket Defect Detection using the TensorFlow open-source library. [9]

However, traditional image processing and machine learning-based methods for O-ring defect inspection methods have some drawbacks which low-accuracy, very time consuming and also high labor intensity. These methods are very sensitive for different sizes of objects which means each model can only inspect a few sizes of O-rings, and can only inspect very limited types of defects that each defect needs to be declared properly (not dynamic). Moreover, software maintenance is difficult that updating the system and bug fixing is time consuming and require domain experts. [5]

In this paper, we proposed a fully DL-based O-ring defect inspection framework that learns the extraction of useful features along with defect classification jointly. The pretrained CNN models have been used by applying the Transfer learning technique. First, we use object detection to determine the location of the image given by a rectangular, axis-parallel bounding box. Then we crop an object one by one and apply some preprocessing techniques such as gamma correction on images of an O-ring to improve their quality. Finally, the classification part defined to extract, from images, features that can distinguish defective and non-defective O-rings and for classifying them into good or not-good.

1.2 Goal

The goal of this thesis work is to find an approach to detect and classify of O-ring defects, with high accuracy, and using modern Deep Learning algorithms. Moreover, we need to increase model robustness which should be able to inspect different size and defect types of O-ring objects. We also examine that what and which stage data preprocessing and postprocessing will improve the result. Our approach should also work in a real-time industrial environment.

1.3 Outline

This thesis begins with an explanation of the operation process of the O-ring Inspection machine and how data is collected. Then in the chapter 3, we will briefly explain the theory of Neural Networks and Convolutional Neural Networks. In the chapter 4, we show our fully deep learning framework and what types of pre-processing methods are used. The chapter 5, Experiments and Result section is divided into two parts. First, we check the overall performance of our model based on test data. In the second part, we compare the effects of different processing measures. Finally, the conclusion will be given in chapter 6.

Chapter 2: DATA

In this section, we present a method used when acquiring the data, and we also present the O-ring Inspection machine which is used to acquire data and classify the O-ring defects in a real-time factory application scenario.

2.1 O-ring Inspection machine

O-ring inspection equipment uses two Grayscale cameras, a PC equipped with a GPU, and a 27-inch monitor. This technology is used to detect surface defects and burrs of O-rings.

The whole process begins when the O-ring is inserted and the equipment is started. Before starting the O-ring shooting process, the O-ring Auto Aligner flattens the O-rings and places them one by one on the camera. Then, the camera shoots continuously at 25.7 FPS per second, inspecting one O-ring about 3-4 times in a 1 frame. The resolution of the camera is 2400×1800, which acquires high-precision images. One of the cameras acquires the image of the top of the O-ring and O-rings will be flipped to the other side and uses the second camera to acquire the image as shown in Figure 1. Even if, one of several images acquired from one O-ring is read as defective, it is judged that there is a defect in the O-ring. When the entire O-ring is not visible over the end, the test result is ignored.

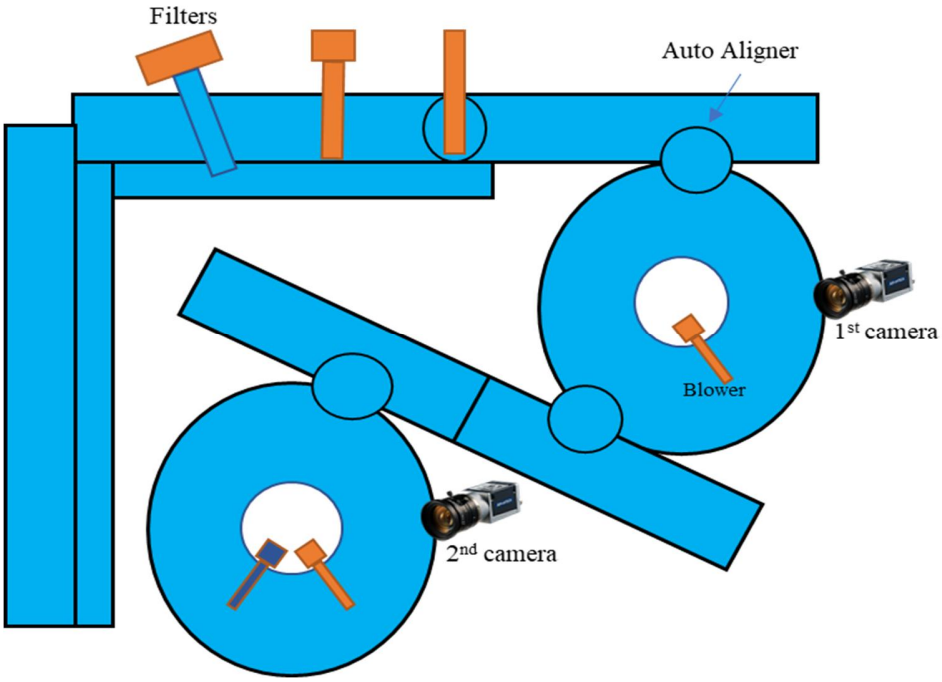


Figure 1. O ring Inspection machine workflow.

2.2 *Data acquisition*

Data was collected in O-ring Inspection project on the factory environment. Generally, the main portion of the gathered data is viewed as training and the other half test data. In the training data, 2439 defective O-rings have been annotated as an "NG" and 1071 non-defective O-rings have been annotated with the name of "OK". On the test data, there are 286 O-ring samples with the label of "NG" and 133 with the label of "OK". A total of 4,321 O-ring images were used in our dataset. A complete description of the final dataset is given in Table 1.

	OK	**NG**
Train	1071	2439
Validation	120	272
Test	133	286
Total	**1324**	**2997**

Table 1. Detailed description of O-ring objects dataset. Good O-ring objects given as OK and not good O-ring objects as NG.

In this thesis, we divide the O-rings into two different categories. Those are “Non-defective” and “defective” O-rings which are shown in Figure 2. Images of these O-rings, taken by the two cameras at the O-ring Inspection machine, are shown in Figure 1.

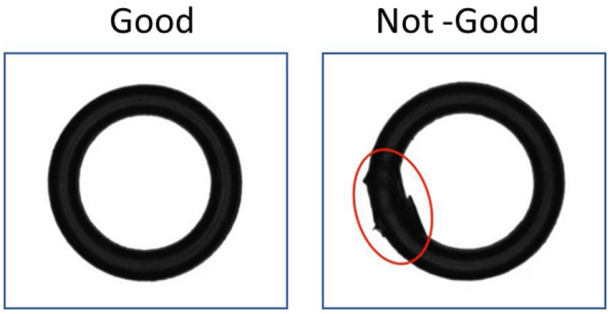


Figure 2. Raw images of the different types of O-rings. The images are captured with the Grayscale cameras at the O-ring inspection machine with 25.7 FPS per second, inspecting one O-ring about 3-4 times in a single frame. Non-defective O-ring (on the left side), defective O-ring (on the right side).

Chapter 3: NEURAL NETWORKS

3.1 The Basics of Neural Networks

A Neural Network is an machine learning algorithm used to make non-linear models for regression and classification. Neural networks are comprised of a number of different functions which are the reason they are called networks. Neural networks are the most utilized deep learning strategy today. In the accompanying section, we present a fundamental structure of neural networks which is known as a perceptron and multilayer perceptron. Then, we explain the various functions and general techniques that we used to build, train and evaluate neural networks. After that, we present CNN, which is the sort of neural network that we used in this thesis work. CNN is usually utilized for image-based computer vision tasks, including defect detection and classification.

3.1.1 Perceptron and Multilayer Perceptron

A perceptron is a unit of the neural network which makes certain computations in the input data to detect or classify the features. It is designed to imitate a neuron, which is modeled afterward the functioning of the human brain. Although the perceptron structure is simple, it is capable to learn and solve even complex problems [10]

The input of x, which is multiplied by the weighted coefficient learned, is represented by the perceptron function and the output value f(x) is generated:

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In the equation given above:

w = vector

b = bias

x = vector of input values

$$\sum_{i=1}^{k} w_i x_i \quad (2)$$

k = perceptron number of inputs.

All inputs of x are multiplied by weights w by the summation ∑ function and applied as follows:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n \tag{3}$$

It is possible to express the output as "1" or "0". It can also be defined as "1" or "-1" based on which activation feature it is. Figure 3 displays the perceptron's input and output:

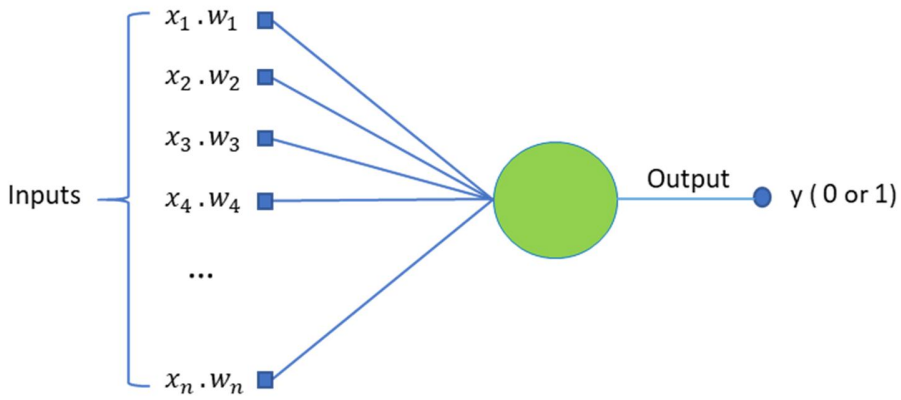


Figure 3. Perceptron Input and Output.

A single layer perceptron can only learn patterns that can be linearly separable. It consists of four main sections, namely values of data, weights, bias, net sum and functions of activation.

Multilayer perceptrons (MLP) are groups of perceptrons composed of many layers that can address specific questions correctly. In the second layer, all the perceptrons in the first layer (on the left) send signals to each of the perceptrons, and the loop goes on like that. The MLP includes, as seen in Figure 4, at least one hidden layer and an output layer in the input layer.

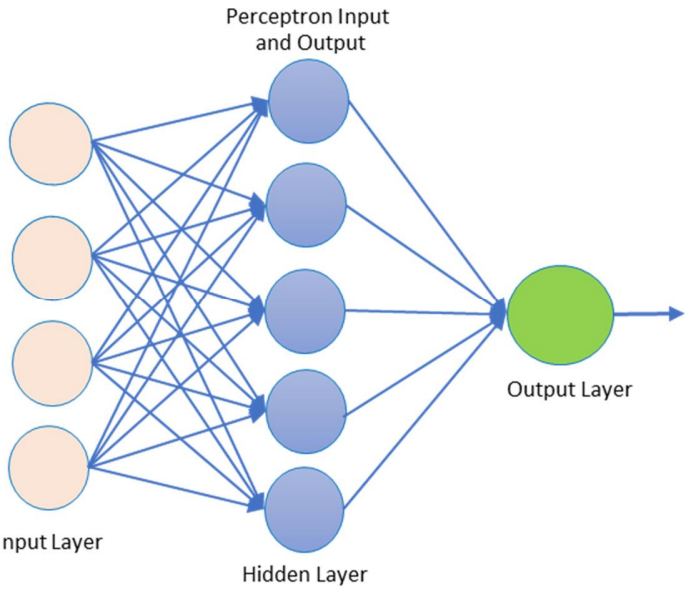


Figure 4. A basic structure of Multilayer Perceptron.

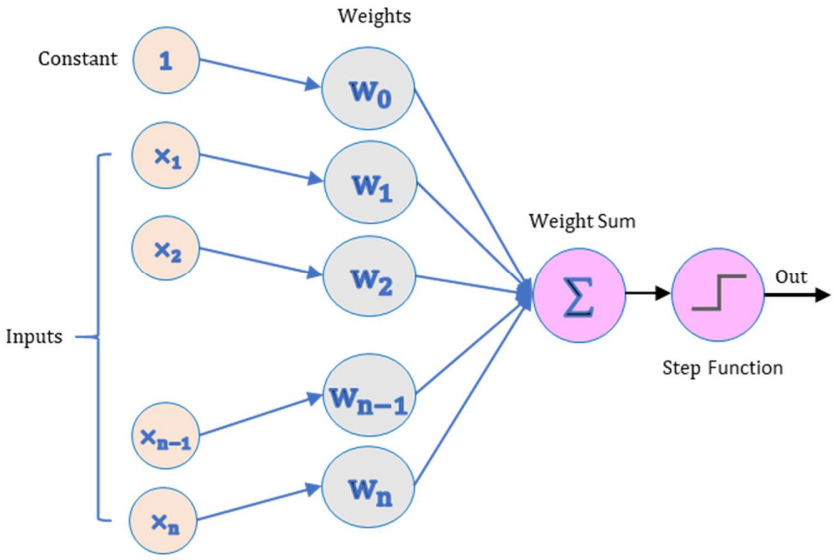


Figure 5. The Perceptron Learning Process.

The perceptron learns as follows:

- Initially, it takes the input supplied to the input layer's perceptron, multiplies it by its weight, then calculates the sum.

- Second, add the number 1 and multiply with "bias weight". It is a technical step that enables to each perceptron's activation function up, output function to be moved up, down, left and right in the numerical graph.
- Third, through the activation function, the sum is fed. The activation function is a step-function in a normal perceptron system.
- The result of the step function is the output.

3.1.2 Activation Functions

The activation function is used to generate nonlinearities or to expand the results in a neural network. Without activation function, a neural network will just be a linear model. Nowadays ReLU is the activation function most mostly used in neural networks, which stands for Rectified Linear Unit *[11]*. ReLU is defined as:

$$f(x) = max(0; x) \tag{4}$$

where *x* is the weighted input to the node, combined with the node specific bias. Another activation function, generally used only neural network's final layer, is the *SoftMax* function. SoftMax is used in many classification algorithms, such as the polynomial logistic regression, to measure all outputs so they are all positive and together sum up to 1. For a class $k, k = 1, \dots K$ in a neural network as in equation 5, the SoftMax function is defined as follows:

$$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_t}} \tag{5}$$

where T_k is the output node for the class *k*. Because SoftMax always has positive outputs which sums to 1, it is possible to view the outputs as probabilities for the observation belonging to each respective class [9].

3.1.3 Regularization

A predictive statistical model ought to have the option to generalize and predict the values or classes of new, has not been trained objects. If the statistical model has very few observations relative The model may run the risk of overfitting with the number of parameters,. When the model is overfitted , it is too well adapted to the training data and cannot effectively classify new observations. Regularization is a restriction on the model in order to reduce overfitting [10]. Early stopping is a common method of regularization when training statistical models. The early stopping method has some rules in order to stop training the model. The rule could be training the network until validation loss function reaches a minimum, or validation

accuracy reaches a maximum, or until a certain determined number of epochs. The one epoch consists of all learning observations that go through the model and the corresponding development.
Batch normalization is a transformation that involves enlarging the input to a button so that the inputs have zero and the average value of the unit variance. Batch normalization makes the model can converge easier and the loss surface smoother. It reduces the risk of a model, to get drawn in towards an unwanted local minimization. Mean and variance of k inputs and the for the batch *B* can be described as:

$$\mu_{Bk} = \frac{1}{b}\sum_{i=1}^{b} \mathcal{T}_{ik} \tag{6}$$

$$\sigma^2{}_{Bk} = \frac{1}{b}\sum_{i=1}^{b} (\mathcal{T}_{ik} - \mu_{Bk})^2 \tag{7}$$

the batch size of b is B and $\mathcal{T}_{ik}$ is the *k*:th input to the node of the *i*:th observation.
For *K* input dimensions, *k = 1…..., K,* and *I = 1,….,b,* we will have that:

$$\hat{\mathcal{T}}_{ik} = \frac{\mathcal{T}_{ik} - \mu_{Bk}}{\sqrt{\sigma^2{}_{Bk}} + \epsilon} \tag{8}$$

where $\hat{\mathcal{T}}_{ik}$ is the *k*:th input of the *i*:th observation after the normalization step. ϵ is a small value to prevent dividing with zero in case of zero-variance batches.

The final step of batch normalization is to transform $\hat{\mathcal{T}}_{ik}$ with the parameters γ_k and ν_k. γ_k and ν_k are learned through backpropagation. The transformation is performed as:

$$\mathcal{Z}_{ik} = \gamma_k \hat{\mathcal{T}}_{ik} + \nu_k \tag{9}$$

$\mathcal{Z}_k$ is the k:th new input for the i:th observation to the node [12].

Dropout is a method of standard regularization it used in the final layers of the neural network. It avoids overfitting by taking away a certain percentage of the nodes from a layer for each batch. Therefore, the network cannot get dependent on certain nodes or a combination of nodes for its prediction and thus the risk of overfitting is decreased.

3.1.4 Backpropagation

Backpropagation is the process of optimizing neural networks. It consists of a "forward pass", where training data is predicted and a loss function is calculated, and a "backward pass", where the network weights are updated and *tuned*. These weights are tuned hierarchically, which means they are first tuned in the output layer, which then tunes the second-last hidden layer, and so on until all layers are tuned.
The loss function, categorical cross entropy, *R(θ),* calculates the entropy between the different classes and is defined as:

$$R(\theta) \equiv \sum_{i=1}^{N} R_i = -\sum_{i=1}^{N}\sum_{k=1}^{K} \mathcal{Y}_{ik} \log(f_k(x_i)) \tag{10}$$

where θ is the complete set of weights, *N* is the number of observations, *K* is the number of classes, $x_i = (x_{i1}, \dots, x_{iP})$ is the *i*:th observation with *P* being the dimension of the input, $\mathcal{Y}_{ik}$ is the true probability distribution of observation *i* belonging to class *k,* and *f* is a classification function for class *k* [12]. R_i is the categorical cross entropy for the observation *i*.

The most common way of tuning the weights of the layers in a neural network is through a *gradient descent* on the loss function. In gradient descent the goal is to reach a minimum through following a function's steepest descent. In Neural networks, we strive to minimize the loss function, in this case the categorical cross-entropy *R(θ),* The steepest descent is calculated separately through partial derivatives for every parameter. The steepest descent is traditionally calculated on the full training set. Calculating the steepest descent on a randomly selected subset of the training set (batch) is called *Stochastic Gradient Descent*. [13]. When working with a big amount of data, the common measure is to use the stochastic gradient descent.

For the same scenario as explained in the two-step neural network as $\beta_k^T = (\beta_{k1}, \beta_{k2}, \dots, \beta_{kM})^T$, and $\alpha_m^T = (\alpha_{m1}, \alpha_{m2}, \dots, \alpha_{mP})^T$, where $m = 1, \dots, M$ and $p = 1, \dots, P$. The partial derivative of the loss function for the parameters β_{km} and α_{mp} for a batch of size *N*, is defined as following:

$$\sum_{i=1}^{N} \frac{\partial R_i}{\partial B_{km}^{(r)}} \tag{11}$$

$$\sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{mp}^{(r)}} \tag{12}$$

The β-parameters are dependent on the α-parameters, so the partial derivative with regards to the α-parameters will be dependent on the partial derivatives of the β -parameters, which is why its called backpropagation. Using the partial derivatives, the formula for updating weights of the parameters β_{km} and α_{mp} are given below in equation 13-14.

$$\beta_{km}^{r+1} = \beta_{km}^{r} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial B_{km}^{(r)}} \tag{13}$$

$$\alpha_{km}^{r+1} = \alpha_{km}^{r} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{mp}^{(r)}} \tag{14}$$

Where r is the iteration step and γ_r is the learning rate, which is decided by the *optimization algorithm*.

Adam, or adaptive moment estimation, is an optimizer of the loss function based on the stochastic gradient descent. It utilizes the stochastic gradient descent but with momentum and adaptive learning rate for all parameters. Adam is today considered a default optimizer when using neural networks**.**

3.1.5 Performance Metrics

Performance metrics are used to evaluate purpose after training the models. There are various evaluation metrics to estimate, the performance of machine learning algorithms. In order to evaluate machine learning performance, we need to choose the metrics carefully because:

- Measured and compared performance of the algorithms for machine learning can depend entirely on the metric that we chose.
- As a result, how we measure the status of different features will have a complete impact on the metrics we choose.

We used various common metrics in this thesis such as : *Confusion Matrix,* Accuracy, *precision*, *recall and F1 score.*

The best way to calculate the output of a classification problem, which may be a class of two or more forms, is by a confusion matrix. The matrix of uncertainty is a two-dimensional table viz. "Actual" and "Anticipated." In comparison, as seen below, all dimensions are 'True Positives' (TP), 'True Negatives' (TN), 'False Positives' (FP), 'False Negatives' (FN):

		Predicted	
		1	**0**
Actual	**1**	True Positive (TP)	False Positive (FP)
	0	False Negative (FP)	True Negative (TN)

Figure 7. Confusion Matrix.

The definition of terms related to the confusion matrix is as follows:

- **True Positives (TP)** − It is the case that both the real data point class and the expected class are 1.
- **True Negatives (TN)** − It is the case that both the real data point class and the expected class are 0.
- **False Positives (FP)** − That is the case where the real data point class is 0 and the expected data point class is 1.
- **False Negatives (FN)** − This is the case where the real data point class is 1 and the expected data point class is 0.

Accuracy is the percentage of correctly classified objects. It is calculated on the full set of predictions [5]. For classification algorithms, it is the most common efficiency metric. As a ratio of all predictions, this can be defined as the amount of correct predictions made. With the aid of the following equation, we can conveniently quantify it by the uncertainty matrix:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad [14]$$

In the search for documents, *precision* is applied, which can be defined as the number of correct documents that our deep learning model returns. With the aid of the following equation, we can easily calculate it via the uncertainty matrix:

$$Precision = \frac{TP}{TP + FP} \quad [14]$$

The recall can be defined by our deep learning model as the number of positive elements recovered. With the aid of the following equation, we can easily calculate it via the uncertainty matrix*:*

$$Recall = \frac{TP}{TP + FN} \quad [14]$$

The *F1 score* would give us the harmonic calculation of accuracy and recall. Mathematically, we should think of the F1 score as the weighted average of precision and recall. The highest F1 value would be 1 and the worst value would be 0. With the help of the following equation, we can compute the F1 score:

$$\boldsymbol{F1 = 2 * (precision * recall) / (precision + recall)} \quad [14]$$

There is an equivalent proportional contribution to precision and recall to the F1 score.

3.2 Convolutional Neural Networks

CNNs are neural networks that conduct convolutional layer feature mappings before using dense layers to classify the object. In deep learning, CNNs have been at the center of developments. Although CNNs were used to solve character recognition tasks as early as the 1990s[16], their present common use is due to much more recent work when using a deep CNN in the ImageNet image classification challenge to beat the state-of-the-art[6].

CNN consists of a large number of neuron layers, each of which is a nonlinear operation in linearly altering the outputs of the anterior layer. The layers mainly include a convolutional and pooling layer. Weights in convolutional layers need to be trained, while pooling layers change the activation using a fixed function.

There are two additional parameters, typically connected to the convolution layer calculation: stride and padding. The stride of the convolution kernel is shifted (step 1. has both height and weight of regular convolution 1. The padding parameter defines how the calculations of the boundary are handled. The first choice is not to do so that step = 1 and convolution kernel 4 × 4 creates outputs smaller by 5 pixels with height and weight. Another alternative is to enable performance or to avoid a possibly big ramp in the signal the border values be the similar values as the nearest pixel.

3.2.1 Convolution

Bread and butter are related transformations of the neural network: a vector is obtained as an input and multiplied by a matrix to produce an output. This applies to any data, whether it's a picture, a sound clip, or a cluttered set of characteristics: their change is always the same no matter what their size. First the vector can be flattened. Pictures, sound clips and lots of other similar kinds of images:

- as multi-dimensional arrays they are stored.
- they have one or more axes where it is important to order
- To access various data views, a single-axis called a channel axis is used

When a transformation is introduced, these properties are not exploited; actually, all the dimensions are viewed in the similar technique and thus the topological knowledge has not been taken into account. However, getting benefit of the tacit knowledge system can prove to be very helpful in solving such tasks, such as machine vision and understanding of voice, and it would be better to maintain it in these instances. It's here where convolutions used for. A convolution is a linear transformation that retains this ordering notion. An instance of a discrete convolution is given in Figure 8. The white grid is called the map of the input function. One input feature map is shown to make the drawing straightforward, although it is not unusual to have several feature maps piled on top of each other A kernel (4 x 4) of value slides through the

map of the input function. The merchandise among to each portion of kernel and hence, an input data overlaps is calculated at each location and the results are summarized to get the output at the current location. Using different kernels, the procedure can be replicated to form as many maps of output features as needed. The final outcomes of this process are called maps of the output function. The kernel will have to be 3-dimensional whether there are several input feature maps, or all feature maps tend to be transformed with a certain kernel equivalently, and then the feature map results will be summarized elementally to provide the output of the feature map.

1	**1**	**-2**	**2**
3	**2**	**1**	**0**
-2	**-3**	**-2**	**2**
0	**1**	**1**	**2**

$$2 \times 1 + 2 \times 1 - 1 \times 2 + 3 \times 2 + 1 \times 3 + 0 \times 2 + 2 \times 1 + 1 \times 0 - 0 \times 2 - 0 \times 3 - 1 \times 2 + 3 \times 2 + 3 \times 0 + 1 \times 1 + 1 \times 1 + 0 \times 2 = 19$$

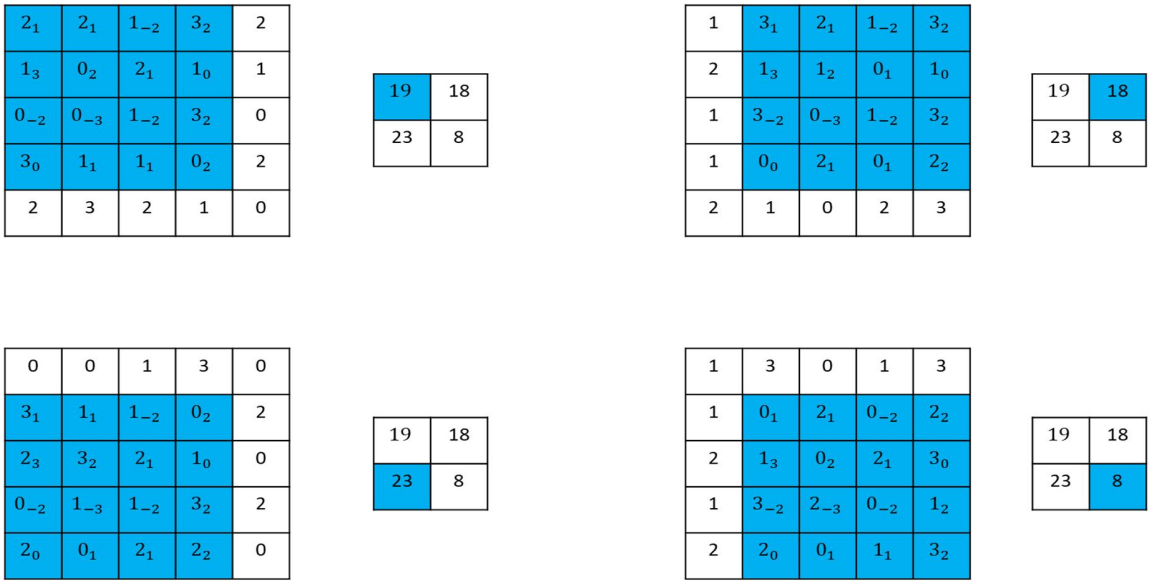


Figure 8. Computing a single convolution's performance values.

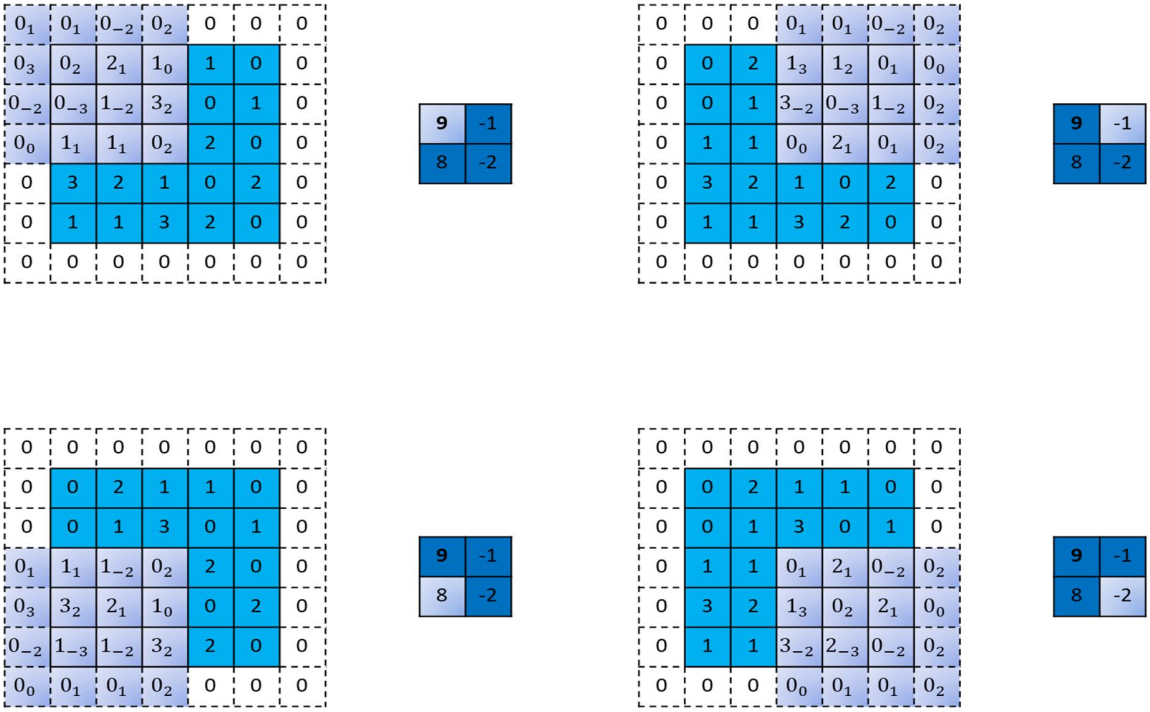


Figure 9. Computing the discrete convolution output values for D = 2, i1 = i2 = 5, k1 = k2 = 4, s1 = s2 = 3, and p1 = p2 = 1.

The example of a 2-D convolution is illustrated in Figure 8, but it tends to be summed up in N-D convolutions in any case. For example, the kernel can be a cuboid during 3-D convolution.

The collection of kernels that define a discrete convolution has a shape such as some permutation of $(m, n, k_1, \ldots, k_N)$, where

$$m \equiv \text{ output of feature maps;}$$

$$n \equiv \text{ input of feature maps;}$$

$$k_j \equiv \text{kernel size}$$

The characteristics influence the output size of a convolutional layer along with axis *j*:

- i_j: input size
- k_j: kernel size
- s_j: stride along with axis *j,*
- p_j: zero padding along with axis *j.*

Figure 9 for example, indicates a 4×4 kernel extended to a 5×5 input padded with a 1×1 zeros using 3×3 strides.

3.2.2 Pooling

Pooling operations form another essential block in CNN, in addition to discrete convolutions. By using a certain function to generalize sub-regions, such as obtaining an average or maximum value, pooling operations minimize the scale of feature maps.

Pooling operates by moving a window over the input and taking control of a pooling function for the content of the window. In some sense, pooling functions just like a separate convolution, except it substitutes the kernel's designated linear combination with some distinct function. For max pooling, Figure 10 shows an example, and Figure 11. For regular pooling, it does the same.

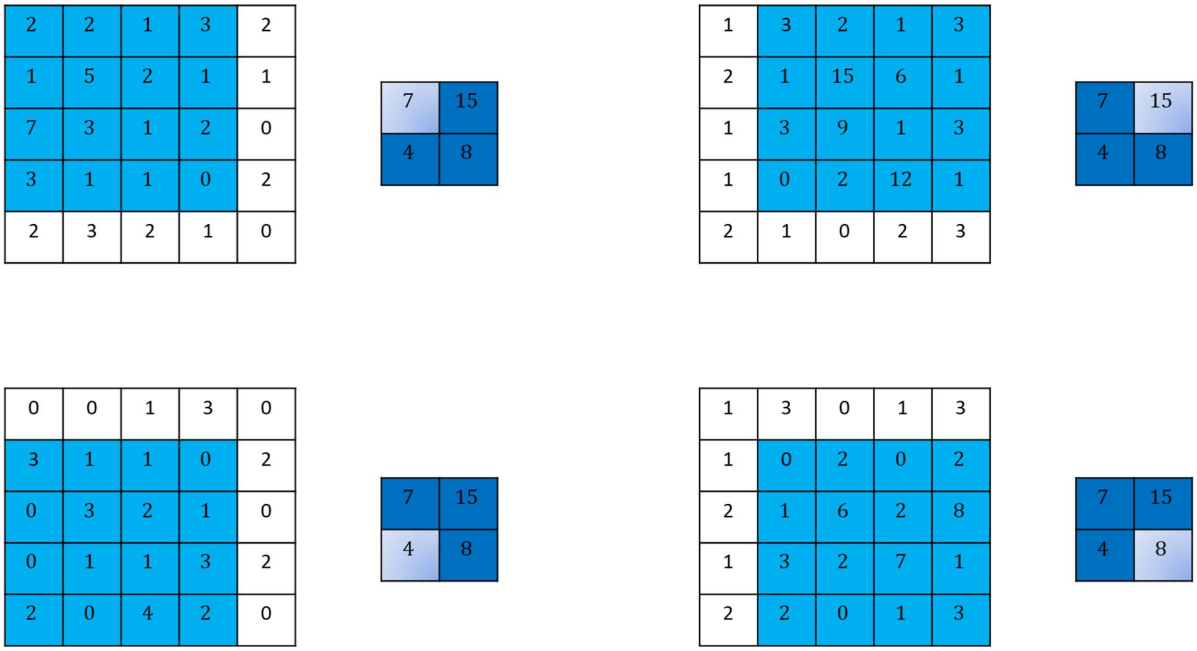


Figure 10. Computing of a 2×2 *max pooling operation on a* 5×5 *input using* 1×1 *strides output values.*

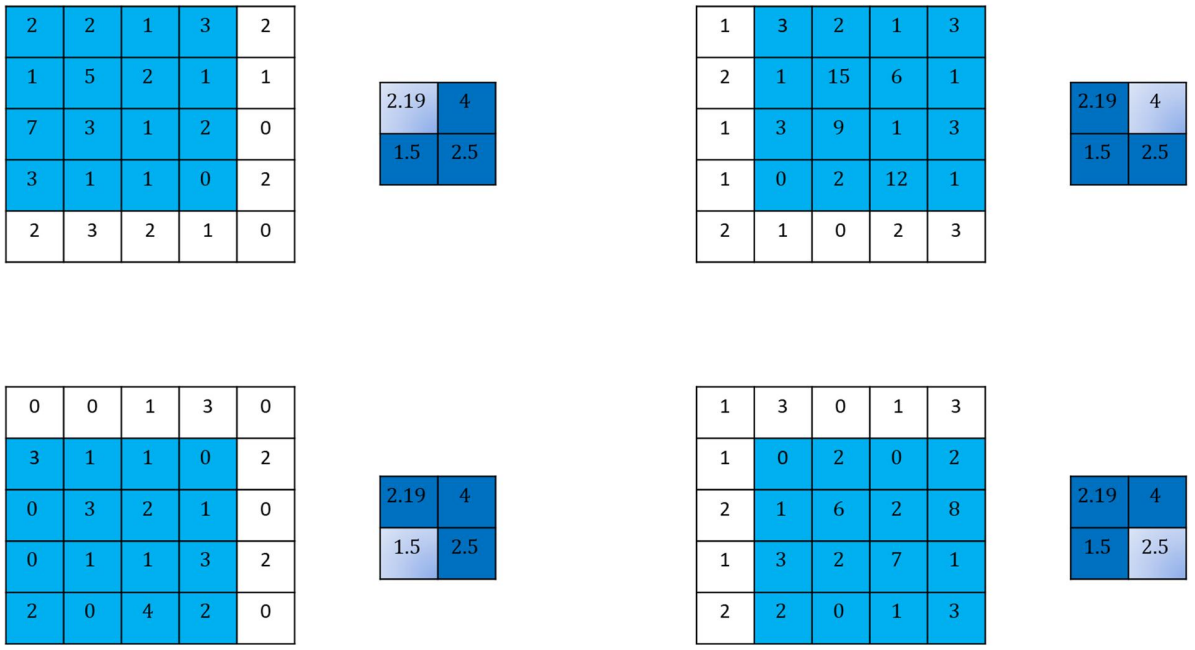


Figure 11. Computing of a 2×2 *average pooling operation on a* 5×5 *input using* 1×1 *strides output values.*

Chapter 4: METHOD

This section describes the methods and programs used in this thesis.

Traditional image processing-based O-ring defect inspection systems first extract hand-crafted features and then passed to a machine-learning algorithm to detect defects [5]. The main problem of traditional image processing-based defect inspections systems is sensitivity to object sizes as well as defect types. Additionally, the maintenance of these kinds of systems is difficult because of the requirement of much effort and time. We proposed a fully DL-based O-ring defect inspection framework that learns the extraction of useful features along with defect classification jointly. An overview of the proposed O-ring defect inspection framework is illustrated in Figure 12.

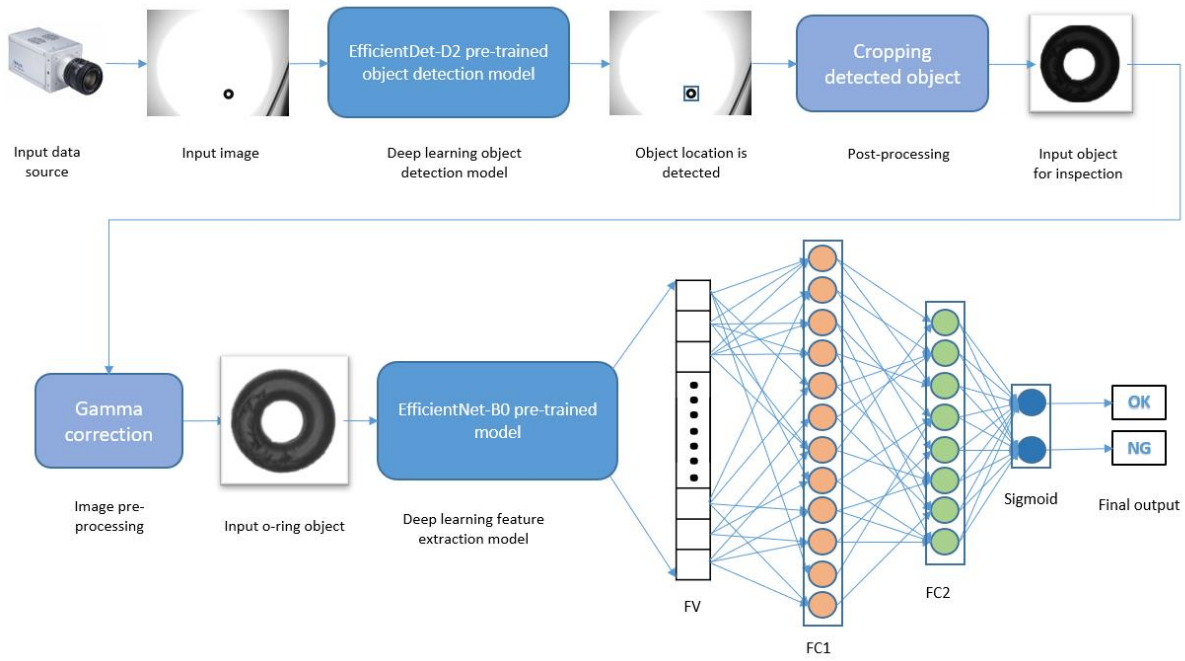


Figure 12. Proposed fully deep learning (DL) based O-ring defect inspection framework. FV - feature vector, FC – fully connected layer.

4.1 Data collection and Data Preprocessing

In this section, we illustrate that how we collected the training and testing data of the models in order to detect and classify O-ring defects. The preprocessing methods that will be evaluated in the Result section are also explained.

If we want to build robust defect inspection system using deep learning (DL) algorithms, we have to fill several DL requirements. For instance, number of training samples should be sufficient in each class for DL model to learn and generalize the solution to problem. There are many public image datasets such as ImageNet [17] for computer vision problems for instance object detection, image classification. However, when it comes to solve computer vision problems similar to object localization and defect inspection in industry, we have to build our own dataset. Dataset was collected via taking pictures from industrial camera which was installed in the O-ring Inspection machine which were presented in section 2.

As DL models require more samples for better accuracy, number of defect samples in each size of O-ring object should be sufficient. Actually, collecting enough number of defective O-ring samples is not easy task than we think. At first, we had very few sizes of defective O-ring data with small number of samples which was insufficient for training process. Hence, to fulfill this requirement, we used 2 types of image augmentation method to increase number of samples in dataset. First one is zooming. In this method we zoomed small sizes of O-ring objects 1.5 and 2 times. The second method is creating artificial defect samples using photoshop program. The second method was applied when number of defect samples were very few in certain size and defect types of O-ring objects. Finally, we have enough number of O-ring samples for training DL object localization and classification models.

4.2 Data cropping and cleaning

After finding the location of the O-ring object, some post-processing steps were applied. Fist one is cropping. The reason is cropping the O-ring objects after detection is to make the task easier for classification model and increasing the robustness.

However, after applying cropping method we need to cleanse the data, because of the industry camera shoots continuously, some frames have more than one object that one of them good and another one is not good, and also some of the objects are semi-defined as they enter and exit the frame. To remove redundant data from the input image, firstly, the system detects the O-ring object location in an input image via using DL object detection model Figure 14 for the detection of the O-ring object in an input image, EfficientDet-D2 [18] pre-trained CNN model was retrained using transfer learning [19] technique. We consider half-cropped objects to be defective so, we manually separate the defective and half-cropped images to the “NG” folder and non-defective objects to the “OK” folder in the training data.

4.3 Gamma correction.

Since it is hard, to not say impossible to see a defect in just one channel, we chose to use multiple channels when classifying a patch to include more information about the surface in the patch. The case of a defect showing in some of the channels but not in others is visualized Figure 13.

Input image comes to the system from a high-quality industry camera that is installed on the top of the O-ring inspection machine. The lighting is given from the bottom side of the O-ring rubber. Thus, the O-ring object looks darker than actually, it should be in an input image. When light is not enough in image then defects will be difficult to analysis and detection. In this case, it is difficult to see and detect defects on the surface of the O-ring objects. To solve this problem, we applied gamma correction [20] in order to fix the lighting condition. The purpose of applying gamma correction is to see defects in images more clearly. Some types of defects which is located on the surface of an object is difficult to recognize when image is dark. So, after gamma correction it can be seen clearly. Classification model also recognize that kind of defects better when gamma transformation is applied before giving to classification model. As shown in Figure 13, O-ring defect on the surface can be seen clearly after applying gamma correction.

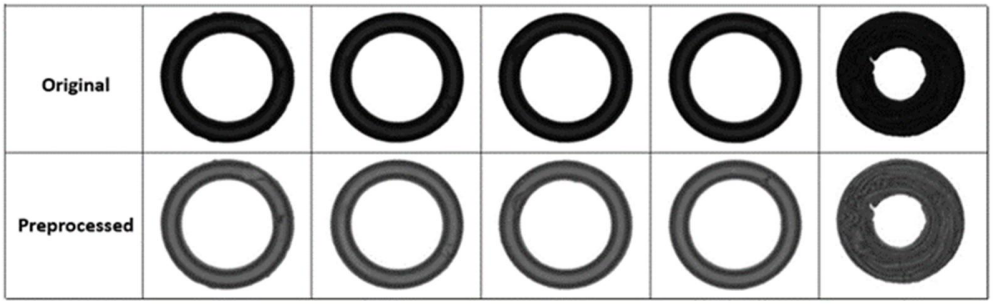


Figure 13. O-ring image samples after applying gamma correction method.

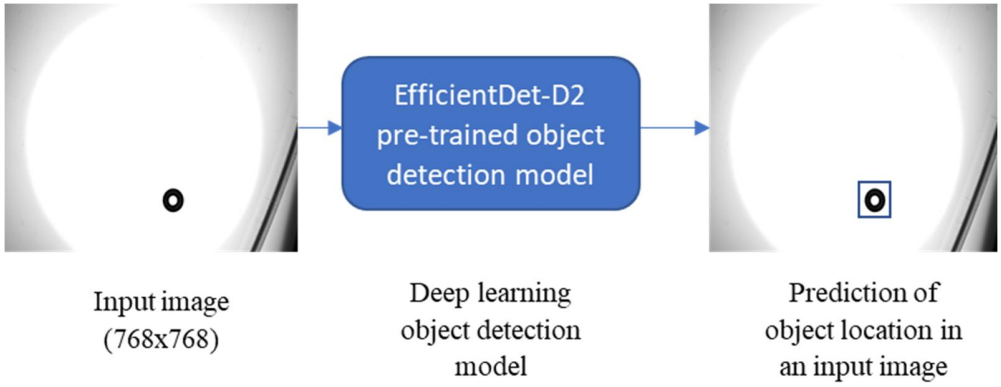
We can control brightness of an image with gamma value. If gamma value is higher than 0.5 then an image will be lighter. If gamma value is less than 0.5 then the processed image will be darker than original image. The formula for calculating the resulting output of gamma correction is given Equation 15. Where I is input image, γ is gamma value to change the brightness, and I' is output image.

$$I' = 255 \times \left(\frac{I}{255}\right)^{\gamma} \tag{15}$$

Since the object detection model performs both object localization and classification, we can use the object detection model for defect inspection task. However, O-ring defects are not easy to inspect, so object

detection model was used only for object localization. Moreover, this increases the robustness of the proposed system to the sizes of different O-ring objects and locations in an input image. General workflow for object detection model is given in Figure 14 .

Figure 14. Object detection model workflow.



The final step is extraction of robust salient features from O-ring input image by using EfficientNet-B0 [21] pre-trained CNN model and classifying it into "*OK*" and "*NG*" classes. Instead of final fully connected layer of EfficientNet-B0 pre-trained CNN model, we used 2 fully connected layer with 128 and 64 hidden nodes respectively. We used transfer learning [19] technique to train DL classification model with modified fully connected layers. The reason to use pre-trained CNN models and transfer learning is that pre-trained CNN models have already been optimized with big dataset to solve general image classification problems. Our dataset was not big, thus using pre-trained models will provide high performance even with small number of samples. The size of feature vector (FV) is 1280, first fully connected layer (FC1) has 128 hidden nodes, and the second fully connected layer (FC2) has 64 hidden nodes. General workflow of DL classification model is given in Figure 15.

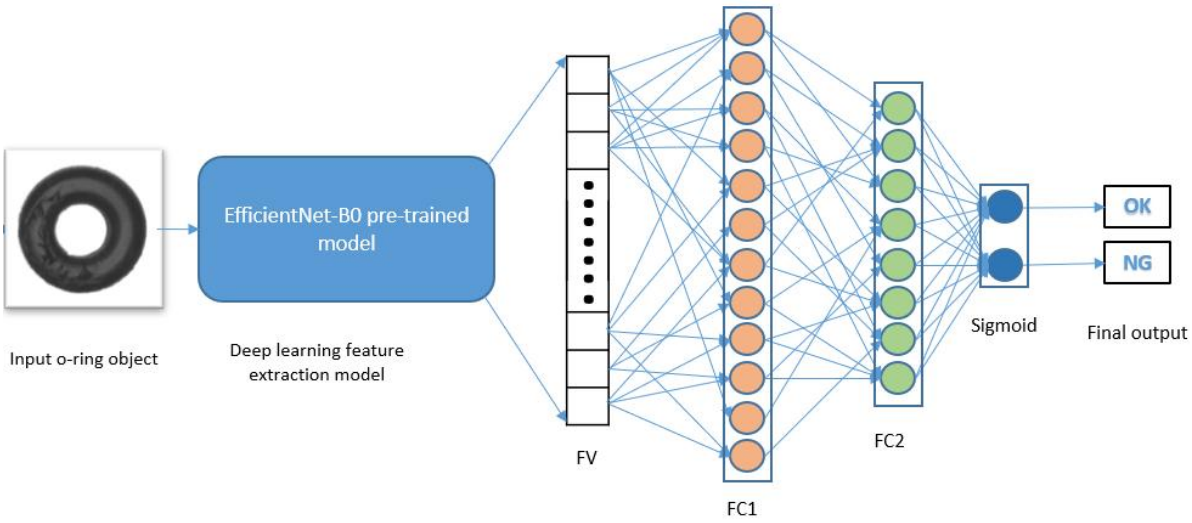


Figure 15. General DL classification model workflow. FV - feature vector, FC – fully connected layer.

Chapter 5: EXPERIMENTS AND RESULTS

5.1 Experimental setup

For DL models training, we used TensorFlow (version 2.2 or above) DL framework and Python (with version 3.7.0) programming language. First of all, we trained DL model for object localization. We used EfficientDet-D2 [10] pre-trained convolutional neural network (CNN) for transfer learning. As object detection model does both object localization and classification, we can use only object detection model for defect inspection. However, O-ring defects are not easy to inspect, so object detection model was used only for object localization. Training parameters for object detection model were set as in the following: batch size was 10, learning rate was 0.0005, and momentum was 0.99. Model was trained on NVIDIA GeForce RTX 2060 GPU which has 6 GB of graphic memory.

EfficientNet-B0 [21] pre-trained CNN model was retrained by changing its last fully connected layer with 2 custom fully connected layers. We applied transfer learning [19] technique to train pre-trained CNN models. Because those models are already trained with big data to solve image classification problem. They are very good for solving general image classification and object detection problems. Additionally, we do not have large dataset to train new CNN model and achieve high accuracy. EfficientNet-B0 [21] pre-trained CNN model gave high classification accuracy after retraining with a few epochs. Training parameters were set as following: batch size was 64, epochs were 50, activation function was RELU, and optimizer was Stochastic Gradient Descent (SGD). The size of feature vector (FV) is 1280, first fully connected layer (FC1) has 128 hidden nodes, and the second fully connected layer (FC2) has 64 hidden nodes. For training 80 % of overall data was used, other 10% data for validation, and the rest 10% was used for testing in both object detection and classification models training.

5.2 Experimental results

Figure 16 demonstrates the result of object detection model on test data. It can be seen from the Figure 16 that localization error is only 1.5 %. As we used DL object detection only for object localization, 98.5 % localization accuracy is quite sufficient for applying this model for finding O-ring objects location in an image in real time.

Figure 16. Object detection result on test data.

Training and validation accuracy result is given in Figure 17. Figure 17 illustrates that DL classification model learned well during training without overfitting and underfitting problems and achieved over 95 % of accuracy for both training and validation data. Training and validation loss result is shown in Figure 18. Both training and validation losses were decreased gradually during optimization of DL classification model. Model gave good classification accuracy on both training and validation data. When DL model gives accurate result on unseen data, then it can be applied to solve real world problem. Testing result of DL classification model is given in Table 3. It can be seen from Table 3 that DL classification model gave

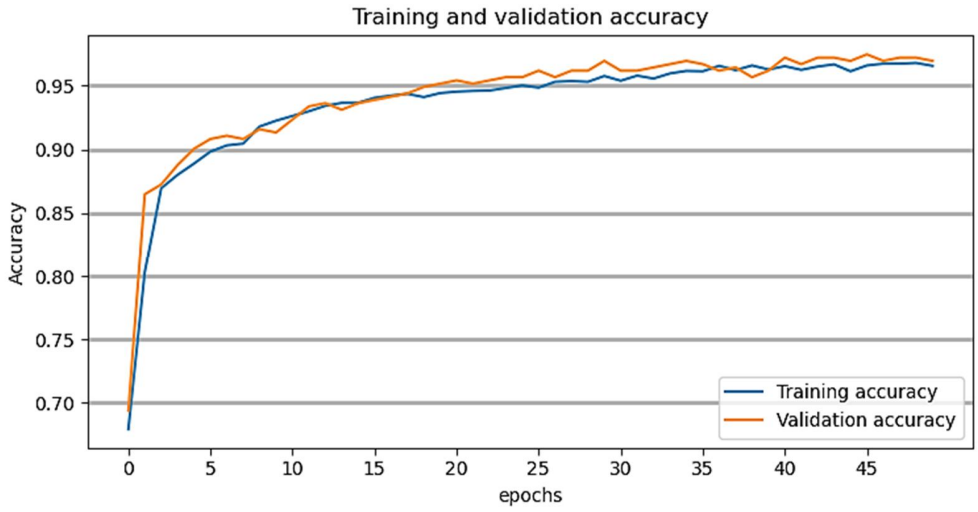


Figure 17. Training and Validation accuracy result during the training

Figure 18. Training and Validation loss result during the training

97 % of accuracy on test data. This means that DL classification model can be applied to solve real world O-ring defect inspection problem.

Table 2. Confusion matrix on a test data.

		Predicted	
		OK	NG
Actual	OK	281	5
	NG	7	126

Precision	Recall	F1	Num of images	Total accuracy
97%	96%	97%	419	**97%**

Table 3. Final testing result of DL classification model.

5.3 Comparing Preprocessing Measures

We compared the different processing measures, by training the models in the O-ring dataset and comparing its performance with the reference model. All of these models use the same structure as the reference model. The performance of the different redesigned models is shown in Table 4 and Figure 19.

Experiments	Confusion Matrix	Accuracy (%)
Non-preprocessing	*Predicted* *Actual* | OK | NG OK | 261 | 25 NG | 34 | 99	85.91
With cropping	*Predicted* *Actual* | OK | NG OK | 263 | 23 NG | 23 | 110	89.02
Cropping & Gamma correction	*Predicted* *Actual* | OK | NG OK | 281 | 5 NG | 7 | 126	97.13

Table 4. Comparison of models excluding different preprocessing measures.

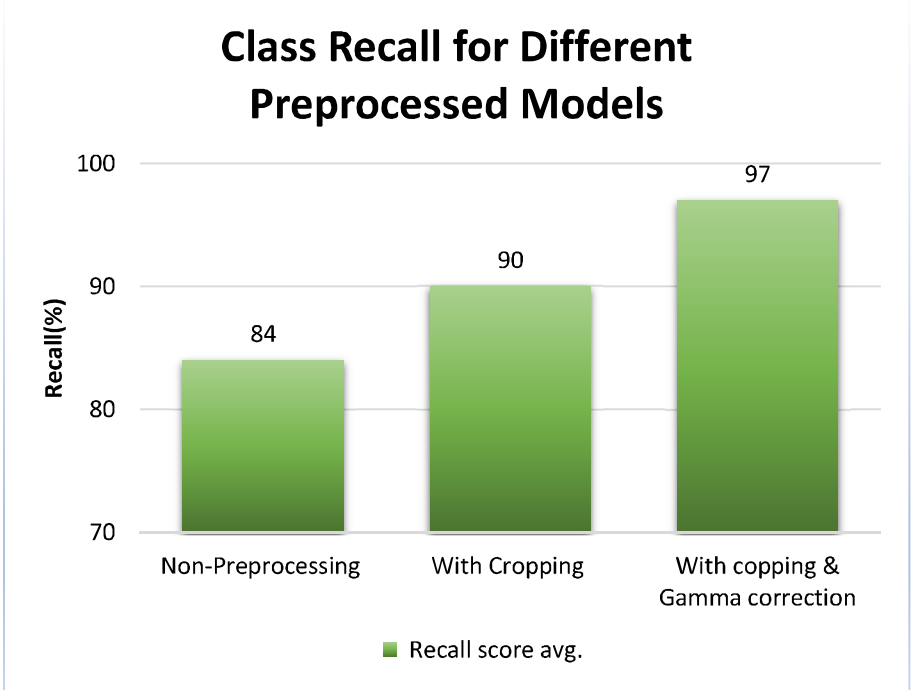


Figure 19. Recall of each class for our differently preprocessed models. Data is based on the confusion matrices in Table 2 and Table. 4.

5.4 Preprocessing Conclusions

Since the original training data set was imbalanced with 1256 non-defective and 714 defective O-ring samples, we created artificial samples by using two augmentation methods, to ensure that training dataset is more balanced to increase accuracy and the ability to distinguish between two classes. Cropping objects and adding gamma correction seems to be the most important measure of pre-processing to achieve higher accuracy. Without any preprocessing, the model still can determine whether O-ring has a defect or not, but it's hard for the model to learn and predict small defects with high accuracy. Therefore, we used such pre-processing measures and achieved a sufficient result.

Chapter 6: CONCLUSION

In this thesis work, we used a custom dataset which images taken from an O-ring inspections machine and annotations made by ourselves. With the development of deep learning methods, we intended to apply them as a method to perform an automated defect inspection. The first goal of this thesis is to find an approach to detect and classify of O-ring defects, with high accuracy, and using modern Deep Learning algorithms such as convolutional neural networks. We also examine that what and which stage data preprocessing and postprocessing will improve the result. And finally, our approach should also work in a real-time industrial environment.

O-ring defect inspection system was built using 2 DL models. Object detection model achieved 98.5 % accuracy on object localization problem. We used transfer learning technique [19] for both object localization and O-ring defect inspection problems. We used EfficientDet-D2 [18] pre-trained CNN model for object localization. For classification problem we used EfficientNet-B0 [21] pre-trained CNN model. After training EfficientNet-B0 [21] CNN model, it achieved 97 % accuracy on test data. From the results, we see that our approach, using CNN models with Transfer Learning technique and preprocessed data, is effective at detecting and classifying defects on the O-ring objects. We have also found that our preprocessing measures have a positive effect on the ability to detect and classify the small defects. The importance of different pre-processing measures is not statistically confirmed, but they all appear to be positive. This means that if no pre-processing measures are taken, the model will perform even worse. Finally, our approach was demonstrated as an achievable approach for a real-time factory environment and together with the accuracy of the classification and the time spent performing these classifications.

REFERENCES

[1] Trelleborg Sealing Solutions "O-Rings" Available:https://www.tss.trelleborg.com/en/products-and-solutions/o-rings, Edition September, 2020.

[2] Martínez, S.S., Ortega, J.G., García, J.G., García, A.S. and Estévez, E.E., An industrial vision system for surface quality inspection of transparent parts. *The International Journal of Advanced Manufacturing Technology*, *68*(5-8), pp.1123-1136, 2013.

[3] Boby, R.A., Sonakar, P.S., Singaperumal, M. and Ramamoorthy, B.,. Identification of defects on highly reflective ring components and analysis using machine vision. *The International Journal of Advanced Manufacturing Technology*, *52*(1-4), pp.217-233, 2011.

[4] Li, Lin, Zhong Wang, and Fang-ying Pei. "Vision-based surface defect inspection for sequence circular objects." In *2012 5th International Congress on Image and Signal Processing*, pp. 1352-1356. IEEE, 2012.

[5] Peng, G., Zhang, Z. and Li, W, Computer vision algorithm for measurement and inspection of O-rings. *Measurement*, *94*, pp.828-836, 2016.

[6] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[7] Masci, Jonathan, Ueli Meier, Dan Ciresan, Jürgen Schmidhuber, and Gabriel Fricout. "Steel defect classification with max-pooling convolutional neural networks." In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-6. IEEE, 2012.

[8] Saiz, Fátima A., Ismael Serrano, Iñigo Barandiarán, and Jairo R. Sánchez. "A Robust and Fast Deep Learning-Based Method for Defect Classification in Steel Surfaces." In *2018 International Conference on Intelligent Systems (IS)*, pp. 455-460. IEEE, 2018.

[9] Chao-Ching, H., Su, E., Li, P.C., Bolger, M.J. and Pan, H.N, Machine Vision and Deep Learning Based Rubber Gasket Defect Detection. *Advances in Technology Innovation*, *5*(2), p.76, 2020.

[10] Gardner, Matt W., and S. R. Dorling. "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences." *Atmospheric environment* 32, no. 14-15, 2627-2636, 1998.

[11] Hara, Kazuyuki, Daisuke Saito, and Hayaru Shouno. "Analysis of function of rectified linear unit used in deep learning." In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8. IEEE, 2015.

[12] Buntine, Wray L., and Andreas S. Weigend. "Bayesian back-propagation." *Complex systems* 5, no. 6, 603-643, 1991.

[13] Ruder, Sebastian. "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747,* 2016.

[14] Goh, Yeow Chong, Xin Qing Cai, Walter Theseira, Giovanni Ko, and Khiam Aik Khor. "Evaluating human versus machine learning performance in classifying research abstracts." *Scientometrics* (2020): 1-16. LeCun, Yann. "LeNet-5, convolutional neural networks." *URL: http://yann. lecun. com/exdb/lenet* 20, no. 5 (2015): 14.

[17] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248-255. IEEE, 2009.

[18] Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781-10790. 2020.

[19] Hussain, Mahbub, Jordan J. Bird, and Diego R. Faria. "A study on CNN transfer learning for image classification." In *UK Workshop on Computational Intelligence*, pp. 191-202. Springer, Cham, 2018.

[20] Rahman, S., Rahman, M.M., Abdullah-Al-Wadud, M., Al-Quaderi, G.D. and Shoyaib, M, An adaptive gamma correction for image enhancement. *EURASIP Journal on Image and Video Processing*, pp.1-13, *(1) 2016.*

[21] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks." *arXiv preprint arXiv:1905.11946* (2019).