A

Project Report

on

Label Inspection System Using Machine Learning

by

Tanawade Hemant Vinayak (X-19-0142)

Padvekar Altamash Yakub (X-19-0185)

Prabhukhanolkar Pragati Madhusudan(X-19-0015)

Bendre Akshay Ashok (X-19-0411)

Under the guidance of

Prof. Sujay. D. Mainkar

Department of Electronics and Telecommunication Engineering

Finolex Academy of Management and Technology, Ratnagiri

2022-23

Finolex Academy of Management and Technology, Ratnagiri

Department of Electronics &Telecommunication Engineering

CERTIFICATE

This is to certify that the project work entitled "**Label Inspection System Using Machine Learning" is a bonafide work of Tanawade Hemant Vinayak, Padvekar Altamash Yakub, Prabhukhanolkar Pragati Madhusudan, Bendre Akshay Ashok** submitted to the Department of Electronics and Telecommunication Engineering, Finolex Academy of Management and Technology, Ratnagiri in partial fulfillment of the requirements for the award of degree of **Bachelor in Electronics and Telecommunication Engineering** as prescribed by the University of Mumbai, in the academic year 2022-2023.

Place: Ratnagiri

Date:

Prof. Sujay D. Mainkar
Guide

Dr. Sharada V. Chougule
HOD EXTC

Dr. Kaushal Prasad
Principal

PROJECT APPROVAL SHEET

Project report entitled as **Label Inspection System Using Machine Learning** presented by **Tanawade Hemant Vinayak, Padvekar Altamash Yakub, Prabhukhanolkar Pragati Madhusudan, Bendre Akshay Ashok** is approved for the degree of ***Bachelor in Electronics and Telecommunication Engineering*** from Finolex Academy of Management and Technology, Ratnagiri.

Prof. Sujay D. Mainkar
Guide

Examiners:

1. _____

2. _____

Date:

Place:

DECLARATION

We declare that this written submission represents our ideas in own words and where others ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We 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.

Name of Student	**Signature**
Tanawade Hemant Vinayak	
Padvekar Altamash Yakub	
Prabhukhanolkar Pragati Madhusudan	
Bendre Akshay Ashok	

Date:

Place:

ACKNOWLEDGEMENT

It gives us an immense pleasure to present the report of our project here. It has been quite experience, facing a number of problems at stages and coming up with appropriate solutions, at time the discussion amongst us or suggestions from our friends and teachers.

We thank our guide Prof. S. D. Mainkar, Assistant Professor Department of Electronics and Telecommunication Engineering, in the best possible way. Without his guidance it wouldn't have been possible to reach this stage. We are very grateful for his support and motivation.

We express our gratitude to Dr. S.V. Chougule, Professor and Head of the Department, Electronics and Telecommunication Engineering for her invaluable suggestions and constant encouragement.

Lastly, we would like to put our thanks on record to the teaching and nonteaching staff for rendering their support directly or indirectly.

Tanawade Hemant Vinayak

Padvekar Altamash Yakub

Prabhukhanolkar Pragati Madhusudan

Bendre Akshay Ashok

Index

Abstract		*i*
List of Figures		*ii*
List of Tables		*iii*
Chapter	Chapter Name	Page
1	Introduction	10
	1.1 Introduction	
	1.2 Motivation	
	1.3 Objective of Project Work	
2	Literature Review	13
3	Overview of System	15
	3.1 Introduction	
	3.2 Methodology	
	3.3 Implementation	
	3.3.1 Product Selection	
	3.3.2 Neural Network	
	3.3.3 Algorithm	
	3.4 Block Diagram	
	3.5 Software	
	3.6 Source Code	
	3.7 Simulation	
4	Conclusion Ans Result	37
	4.1 Result	
	4.2 Future Scope	
	4.3 Conclusion	
	Reference	41

Abstract

Product labeling is an important aspect of the manufacturing process, as it provides information to consumers about the contents, usage, and safety of the product. Incorrect or missing labeling can lead to confusion, legal issues, and health risks. In this project, we propose a label inspection system using machine learning that can automatically detect and classify incorrect labeling. The proposed system involves acquiring and pre-processing images of product labels, extracting and selecting relevant features, and training a machine learning model, such as a convolutional neural network (CNN), to classify the labeling as correct or incorrect. The trained model is then tested and evaluated using a separate dataset, and once deployed, it can analyze new images of product labels in real-time and flag any errors or defects in the labeling. The label inspection system using machine learning can improve accuracy and efficiency in label inspection and verification, reduce errors and costs, and ensure compliance with regulatory standards.

List of Figures

Figure No.	Figure	Page
Fig 3.1	Sample Product	16
Fig 3.2	Biological Neural Network	17
Fig 3.3	Procedure of Reading Images by Computer	18
Fig 3.4	CNN Architecture	19
Fig 3.5	Image Stored in Computer and Karnel	19
Fig 3.6	Convolution process	20
Fig 3.7	ReLU Activation Function	21
Fig 3.8	Max Pooling Layer	22
Fig 3.9	Flattening	23
Fig 3.10	Fully Connected Layer	23
Fig 3.11	Block Diagram	24
Fig 3.12	Python	25
Fig 3.13	SKlearn	26
Fig 3.14	TensorFlow	26
Fig 3.15	Keras	27
Fig 3.16	OpenCV	28
Fig 3.17	Dataset Directory Where We Stored Labeled Dataset	33
Fig 3.18	Detecting Porper Product	33
Fig 3.19	Detecting Defective Product	34
Fig 3.20	Classification Report	34
Fig 3.21	System is Recognizing as a Ok Product	36
Fig 3.22	System is Recognizing as a Defective Product	36
Fig 4.1	Ok Front Image	37
Fig 4.2	Prediction for Ok Product	37
Fig 4.3	Defective Front Image	38
Fig 4.4	Predicted for Defective Product	38

List of Tables

Table No.	Table	Page
Table 3.1	Classification report	35

Chapter 1

Introduction

1.1 Introduction:

A label inspection system using machine learning is a system that utilizes machine learning algorithms to automate the process of inspecting and verifying labels on products or packages. This system is typically used in manufacturing or logistics settings where accurate labeling is critical for regulatory compliance, safety, and consumer satisfaction.

Traditional label inspection systems use rule-based methods to detect errors such as missing or misprinted labels. However, these systems are often limited in their ability to handle variations in labeling due to factors such as different languages, fonts, and label placements.

A label inspection system that employs machine learning algorithms, on the other hand, can learn from a large dataset of labeled images to recognize patterns and variations in labeling. This allows the system to accurately identify and flag errors in labeling, even when the labels are non-standard or complex.

The label inspection system using machine learning typically involves the following steps:

1.Collection of labeled images of products or packages

2.Training of machine learning algorithms using the labeled images

3.Deployment of the trained model in a label inspection system

4.Integration of the label inspection system into the manufacturing or logistics process for real-time label inspection and verification.

Overall, a label inspection system using machine learning can improve labeling accuracy, reduce the need for manual inspection, and increase efficiency in manufacturing and logistics operations.

1.2 Motivation:

There are several motivations behind the development of a label inspection system using machine learning:

1. Accuracy: Accurate labeling is critical for regulatory compliance, safety, and consumer satisfaction. Traditional label inspection systems that use rule-based methods may miss errors in labeling or produce false positives, leading to inaccurate

results. Machine learning algorithms can learn from a large dataset of labeled images to recognize patterns and variations in labeling, resulting in more accurate labeling verification.

2. Efficiency: In a manufacturing or logistics setting, manual inspection of labels can be time-consuming and costly. A label inspection system using machine learning can automate the process of label inspection, resulting in faster and more efficient operations.
3. Adaptability: Labeling can vary based on factors such as language, font, and label placement. Traditional label inspection systems may struggle to handle this variability, leading to errors in inspection. A label inspection system using machine learning can adapt to variations in labeling and learn to identify patterns and variations in labeling, resulting in better inspection results.
4. Scalability: As the volume of products or packages to be inspected increases, traditional label inspection methods may become overwhelmed. A label inspection system using machine learning can handle a large volume of images and can be easily scaled to accommodate increased volumes of products or packages.

Overall, a label inspection system using machine learning can improve accuracy, efficiency, adaptability, and scalability in label inspection and verification, making it a valuable tool for manufacturing and logistics operation

1.3 Objective of Project Work:

Design, develop, and deploy a system that can automate the process of label inspection and verification for products or packages in a manufacturing or logistics setting. The project work can include the following objectives:

1. Data collection: Collecting a dataset of labeled images of products or packages that will be used to train the machine learning model.
2. Data preparation: Pre-processing and cleaning the dataset to ensure it is suitable for use in training the machine learning model.
3. Machine learning model selection: Selecting the appropriate machine learning algorithms and techniques to train the model on the labeled image dataset.
4. Model training: Training the machine learning model using the labeled image dataset to learn patterns and variations in labeling.
5. Model evaluation: Evaluating the performance of the trained machine learning model

on a validation dataset to ensure it can accurately detect and flag errors in labeling.

6. Model deployment: Deploying the trained machine learning model in a label inspection system that can be integrated into the manufacturing or logistics process for real-time label inspection and verification.
7. Performance monitoring: Monitoring the performance of the label inspection system and making necessary adjustments to improve accuracy, efficiency, and scalability.

The ultimate objective of a label inspection system using machine learning project work is to develop a system that can improve accuracy, efficiency, adaptability, and scalability in label inspection and verification, making it a valuable tool for manufacturing and logistics operations.

Chapter 2

Literature Review

A literature review on a label inspection system using machine learning would typically include an overview of existing label inspection systems, their limitations, and the advantages of using machine learning for label inspection. Some relevant literature on this topic is:

1. "A comprehensive review on automated visual inspection systems for quality control of products" by S. Suresh and S. N. Omkar (2018). This review provides an overview of automated visual inspection systems, including label inspection systems, and discusses the advantages and limitations of different types of systems. The authors highlight the importance of accurate label inspection in quality control and discuss the potential for using machine learning algorithms for label inspection.
2. "An intelligent vision system for label inspection using convolutional neural networks" by S. Zhao et al. (2020). This paper proposes a label inspection system using convolutional neural networks (CNNs) for detecting and classifying label defects in real-time. The authors report high accuracy and efficiency of their system and suggest that it can be easily integrated into manufacturing or logistics processes.
3. "A deep learning-based approach for label inspection in pharmaceutical industry" by A. V. Bongale and A. G. Keskar (2020). This paper proposes a deep learning-based approach for label inspection in the pharmaceutical industry, using a combination of CNNs and transfer learning. The authors report high accuracy and efficiency of their approach and suggest that it can be applied to other industries as well.
4. "Label detection and recognition in logistics using deep learning" by H. Zheng et al. (2019). This paper proposes a label detection and recognition system using deep learning algorithms for logistics applications. The authors report high accuracy and efficiency of their system and suggest that it can be used for real-time label inspection and verification in logistics processes.
5. Traian Rebedea “Expiry date recognition using deep neural networks,”, January 2020. conducted a survey a deep learning solution for optical character recogintion, while Yu Wu1 (2019)[14] compare the efficacy of browsing, tagging, and hybrid procedures. However, both reviews have a narrow focus by solely considering the organisation of labelling and the impact of user interface designs.

6. N. Mohd Saad (2013)[2] provide an overview of content-based image retrieval with a strong focus on image search but also cover medical labelling software.
7. T.Ananthan (2015)[3] review automatic image labelling techniques focusing on image retrieval Evgenicy krasnakutshy et al. (2021)[5] conducted a survey of semantic image and video annotation soft-ware and systematise the labelling process by developing a general input-output model and defining categories for various annotation levels. They have not updated their survey since 2019.
8. Ch Anush (2021)[7] review the state-of-the-art in video annotation. In summary, none of the surveys provides a current view or was conducted from an application and domain-independent perspective covering chiefly the task of image labelling or Rohit bhisey (2018)[13] review the state-of-the-art in video annotation. In summary, none of the surveys provides a current view or was conducted from an application and domain-independent perspective covering chiefly the task of ILS (image labelling system).
9. "Automated inspection system for food packaging based on machine vision and deep learning" by Y. Jia et al. (2021): This paper presents an automated inspection system for food packaging that uses machine vision and deep learning to detect defects in the packaging labels.
10. "Real-time intelligent label inspection system based on convolutional neural networks" by X. Cheng et al. (2020): This paper proposes a real-time intelligent label inspection system that uses convolutional neural networks to detect label defects in real-time, providing quick feedback to operators.
11. "Deep learning-based defect detection in printed circuit boards using extreme learning machine classifier" by P. Ramesh and V. Santhosh Kumar (2021): This paper proposes a deep learning-based defect detection system for printed circuit boards using an extreme learning machine classifier.

Overall, the literature suggests that label inspection systems using machine learning algorithms, such as CNNs and deep learning, have the potential to improve accuracy, efficiency, and adaptability in label inspection and verification for various industries. However, further research is needed to explore the limitations and challenges of these systems, as well as their scalability and performance in real-world manufacturing and logistics settings.

Chapter 3

Overview of system and software development

3.1 Introduction:

Convolutional neural networks (CNNs) are a type of deep learning algorithm that has been successfully applied to image recognition, classification, and segmentation tasks. CNNs are particularly effective for image-based tasks because they are designed to recognize patterns and features in images through a hierarchical learning process.

In the context of a label inspection system using machine learning, CNNs can be used to analyze images of product labels and detect any defects or errors in the labeling. The CNN can be trained on a dataset of labeled images, where each image is labeled as either correct or incorrect. During training, the CNN learns to recognize patterns and features in the labeled images that distinguish correct from incorrect labeling.

Once the CNN is trained, it can be used to analyze new images of product labels in real-time and flag any errors or defects. The CNN can be integrated into a label inspection system, where it can automatically inspect and verify the labels on products or packages, improving accuracy and efficiency in the manufacturing or logistics process.

Overall, CNNs have shown great potential in label inspection systems using machine learning, as they can learn complex patterns and features in images and improve accuracy and efficiency in label inspection and verification. However, the effectiveness of the CNN depends on the quality and size of the training dataset, as well as the complexity and variability of the labeling patterns.

3.2 Methodology:

We have collected different product of images and stored in a directory. We have preprocessed the dataset by normalizing, resizing and cropping the images. We have classified dataset into training and testing sets to evaluate the performance of the machine learning model and prevent overfitting. The training dataset is used to train the machine learning model, while the testing dataset is used to evaluate its accuracy and efficiency.

when we fit our data to the model it will Extract relevant features from the images, such as colour histograms, texture analysis, and edge detection and it will Select the most relevant

and informative features using feature selection techniques. here we have selected convolutional neural network based on the characteristics of the dataset and the problem domain. we have trained the selected machine learning algorithm on the training dataset using the selected features. then Evaluated the performance of the trained machine learning model on the testing dataset using metrics such as accuracy, precision, recall, and F1-score.After that we found different between ideal output and actual output. Difference between ideal output and actual output is equal to be the error. There we have found error in our model. If the error is less than our model will be more efficient. After this our model will be ready for prediction.

3.3 Implementation of methodology:

3.3.1 Product Selection:

Fig 3.1 Sample Product

1. Food and beverage products: Labels on food and beverage products often contain important information such as ingredients, nutritional value, and expiration dates. A label inspection system can be used to ensure that the correct information is displayed on the label and that it is legible.
2. Medical products: Labels on medical products such as prescription medications, over-the-counter drugs, and medical devices often contain important safety information. A label inspection system can be used to ensure that the information is accurate and legible.

3. Consumer products: Labels on consumer products such as cleaning products, cosmetics, and electronics often contain important information such as usage instructions, warnings, and safety information. A label inspection system can be used to ensure that the information is accurate and legible.
4. Industrial products: Labels on industrial products such as chemicals, machinery, and equipment often contain important safety information. A label inspection system can be used to ensure that the information is accurate and legible.

Overall, a label inspection system using machine learning can be applied to a wide variety of products and labels to ensure that the correct information is displayed and that it is legible.

3.3.2 Neural Network:

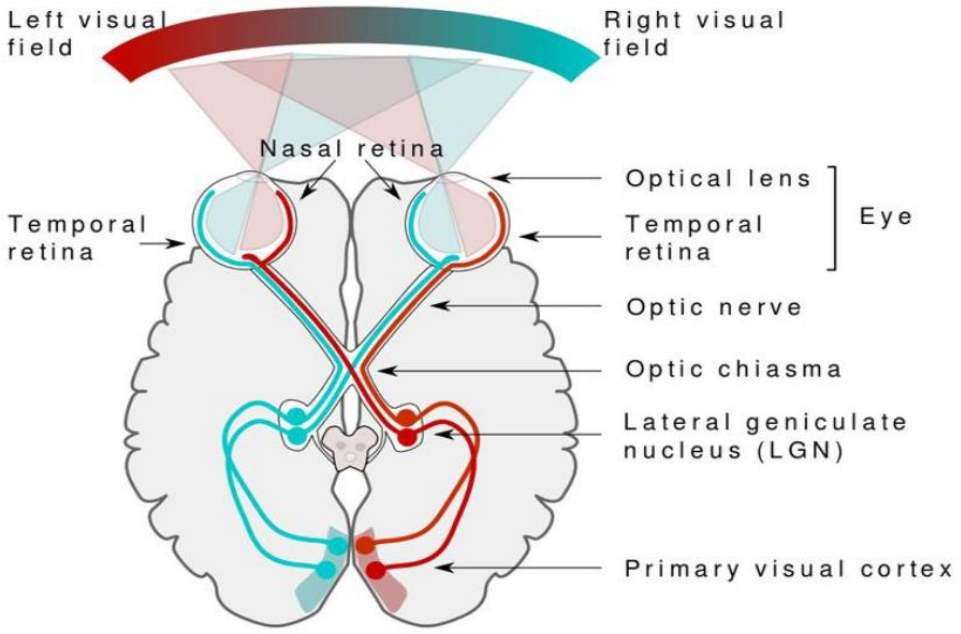

Fig 3.2 Biological Neural Network

Lights and colours on the retina are captured in our eyes. These signals cross the receptors of the retina to the optical nerve and are transferred to the brain for this information to be meaningful. The eye and the visual brain are extremely complex and hierarchical. In perceiving and understanding what we around us view, the full visual path plays a vital part. It is this mechanism inside of us that allows us to understand the image above, the text of this essay, and every other work we do daily. So these processes we are doing from our childhood and so we can predict and analyze through eyes and brain coordination.

So the same process, is that possible for computers? and the simple answer is YES!. computers can SEE the world but in a different way – through numbers. Like the picture below,

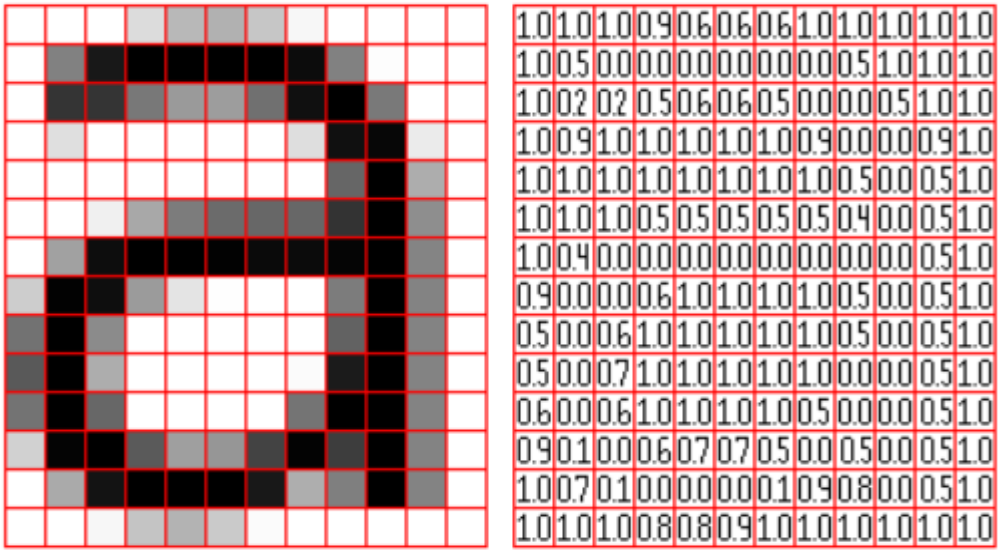

Fig 3.3 Procedure of Reading Images by Computer

Computers understand images by representing them as digital data, typically in the form of pixels. Each pixel represents a small unit of the image and has a numerical value that corresponds to its brightness or colour. The computer then processes this digital data using algorithms that extract meaningful features from the image.

3.3.3 Algorithm:

In Convolutional Neural Network (CNN) each layer concentrates each pattern and its importance like firstly it observes simple patterns such as lines, curves, then some complex patterns like texts, objects, and face and more complex patterns present in the image.
So there are various steps to achieve this CNN algorithm, they are as follows,

1. Convolution layer
2. Activation function (ReLu layer)
3. Pooling
4. Flattening
5. Full connection

On the whole, the below image will explain how CNN works,

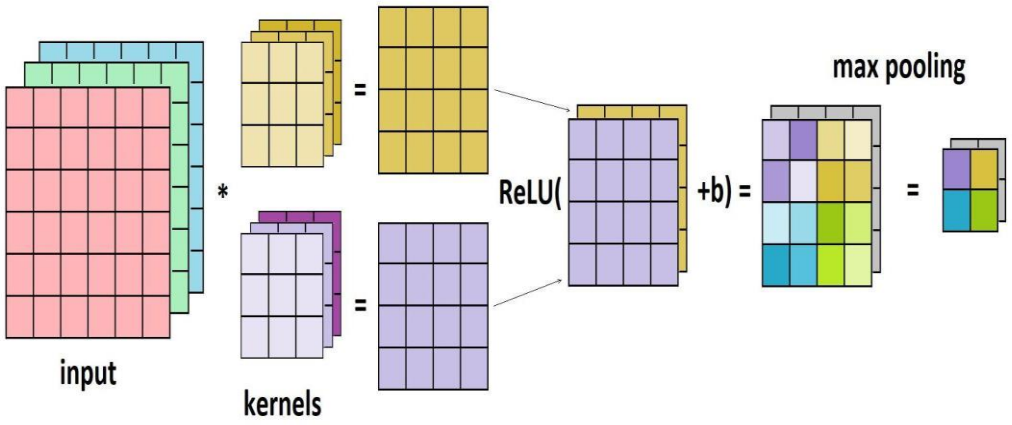

Fig 3.4 CNN Architecture

1. Convolution Layer:

The most important portion or segment in the CNN algorithm. The mathematical notation of Convolution is as follows,

$$(\boldsymbol{f} * \boldsymbol{g})(\boldsymbol{t}) = \int_{-\infty}^{\infty} \boldsymbol{f}(\boldsymbol{\tau})\boldsymbol{g}(\boldsymbol{t}-\boldsymbol{\tau})\boldsymbol{d\tau}$$

here in the above equation, we can infer that convolution is an integration of two simple functions (f and g). We have two terms, one is our input/dataset in matrix format and another one is feature detector or kernel or filters (it will be 3*3 or 5*5 or 7*7 matrix, depends on the architecture that we are using!) when compared to the input image matrix, this matrix is smaller but we can get deeper knowledge from this kernel part only.

Consider an input and kernel matrix as

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

1	0	1
0	1	0
1	0	1

Fig 3.5 Image Stored in Computer and Karnel

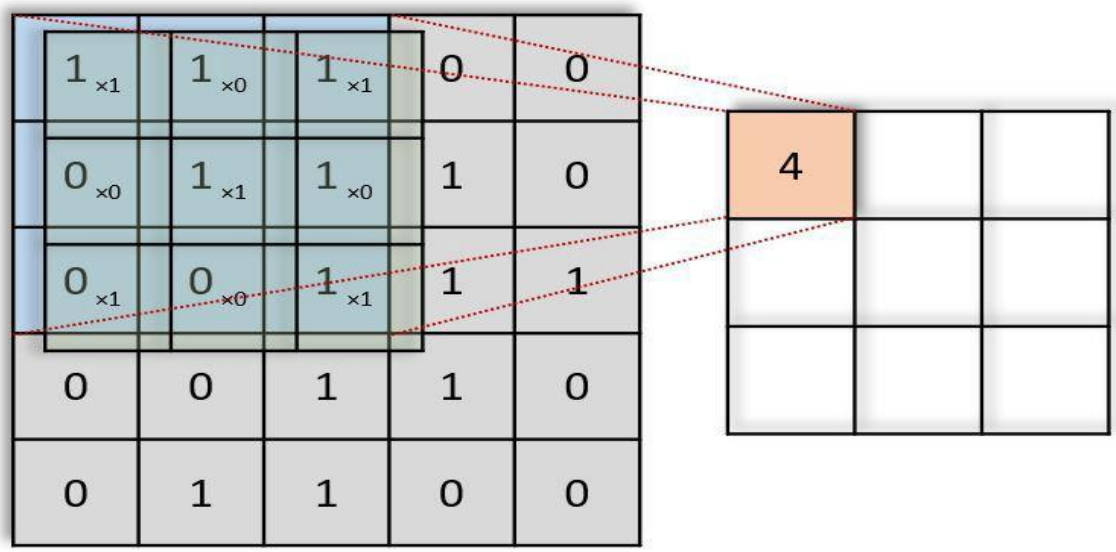

Fig 3.6 Convolution Process

In this above fig, the value 4 in the output matrix, which depends on the 9 values of 3*3 matrix

and another importance is the value doesn't change if any position of the value of the input matrix changes too.

2. Activation Function:

So once the convolution process gets over for our input dataset, the next important step to proceed in the CNN algorithm is the activation function.

When comparing with a neuron-based model that is in our brains, the activation function is at the end deciding what is to be fired to the next neuron. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell. It's a non-linear transformation that we do for inputs that we receive before sending them to the next layer or neuron. AF is also known as Transfer Function.

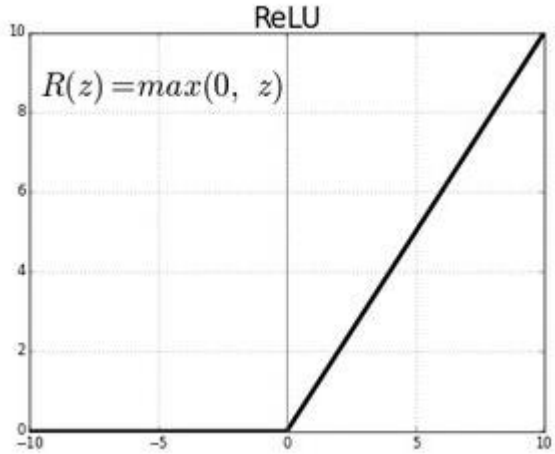

Fig 3.7 ReLU Activation Function

The Rectified Linear Unit (ReLU) activation function is commonly used in convolutional neural networks (CNNs) because of its simplicity and effectiveness in promoting and efficient representations.

The ReLU function takes an input x and return the maximum of 0 and x, as shown in equation: F(x)=max(0,x)

This means that if the input x is negative, the function return 0, and if x is positive, the function return x. The ReLU function introduces non-linearity into the output of a neuron, which is important for the network to learn complex representations.

In a CNN, ReLU is typically applied after the convolutional layer and before the pooling layer. The purpose of the ReLU activation function is to introduce non-linearity into the model, which allows the network to learn more complex features and make better predictions.

Overall, the ReLU activation function is a popular choice for CNNs due to its simplicity, effectiveness, and ability to address some common problems in deep learning.

3.Pooling:

The main importance of the pooling layer is to detect the edges, corners and in application-like images, it can be used to detect facial features like nose, eyes by using multiple filters. It's like filtering the feature map, because we get the feature map from the convolution process, and here in pooling we are going to apply it to the filter. Mostly the size of the filter is 2*2,

which means in the pooling process feature map gets reduced by the factor of 2 (in simple reduced by half of the feature map dimension), intern it reduces each value in the feature map to one quarter.

The main importance of pooling is to reduce overfitting and computation in our dataset.

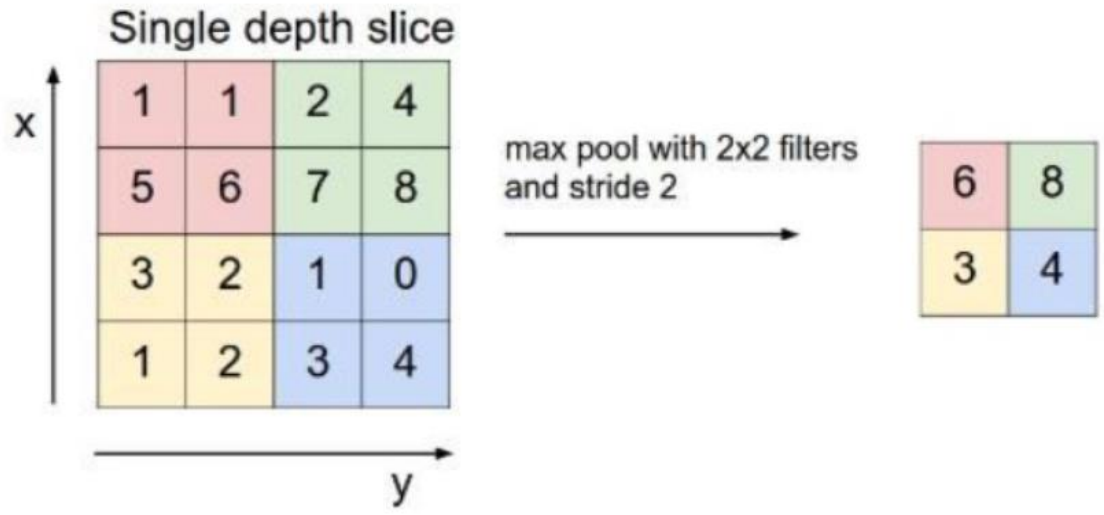

Fig 3.8 Max Pooling Layer

There are some important terms to be noticed in the Pooling process,

1. Filter Size – Describes filters size like 2*2 or 3*3 or anything

2. Stride – describes the number of steps filter takes while traversing the image

3. Padding – Sometimes filter size creates a border effect in the feature map, the effect can be overcome through padding.

4.Flattening:

Flattening is an important step in Convolutional Neural Networks (CNNs) that is used to transform the output of a convolutional layer into a 1D vector that can be fed into a fully connected layer.

In a typical CNN architecture, the input to the network is a 3D tensor of shape (batch_size, height, width, channels), where batch_size is the number of examples in each batch, height and width are the dimensions of the input image, and channels represents the number of color channels in the input image (e.g., 3 for RGB images).

The flattened output serves as the input to the fully connected layers of the CNN, which perform classification or regression tasks on the input image. The fully connected layers have a 1D input shape, and the flattened output from the convolutional layers provides a compatible input for the fully connected layers.

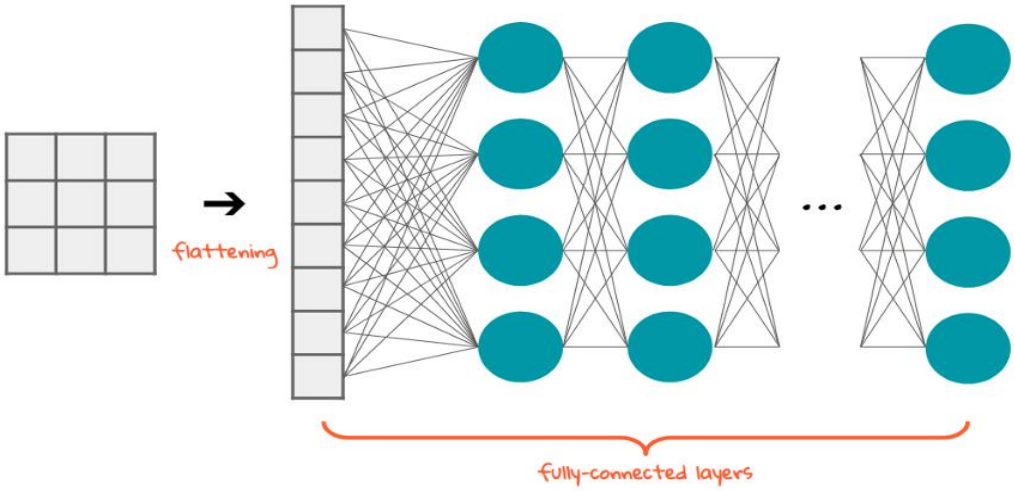

Fig 3.9 Flattening

Overall, flattening is an important step in CNNs as it enables the convolutional layers to be followed by fully connected layers, which can perform complex classification or regression tasks.

5.Fully Connected Layer:

Last layer, in which each pixel or each number/point is considered as a separate neuron just like a NN. The last fully-connected layer will contain as many neurons as the number of classes to be predicted.

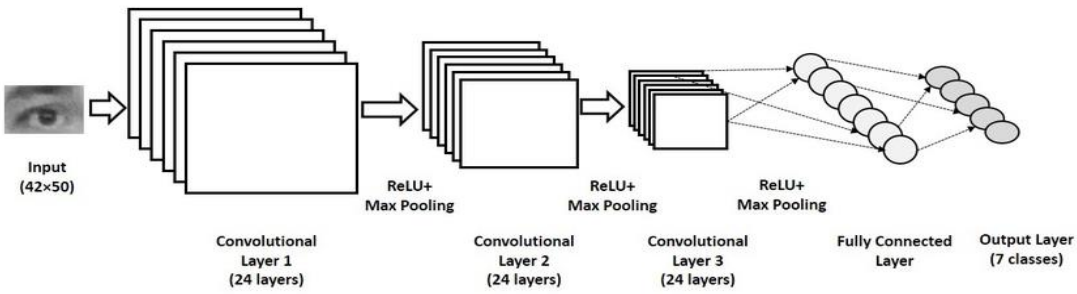

Fig 3.10 Fully Connected Layer

Now that we have converted our input image into a suitable form for our Multi-Level fully connected architecture, we shall flatten the image into one column vector. The flattened output is fed to a feed-forward neural network and backpropagation is applied to every iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them.

3.4 Block diagram:

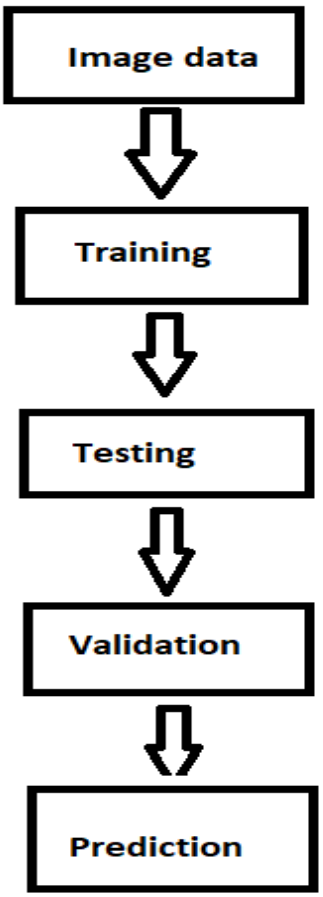

Fig 3.11 Block Diagram

Description**:**

We have collected images and stored in a directory. We have classified dataset into two part like training set and testing set. Suppose that if we have 2000 images then we have to use 1600 images for training purpose and 400 images for testing purpose. After that we found different between ideal output and actual output. Difference between ideal output and actual

output is equal to the error. There we have found error in our model. If the error is less than our model will more efficient. After testing we have set probability (0.5) for equal identification. After this our model will be ready for prediction.

3.5 Software:

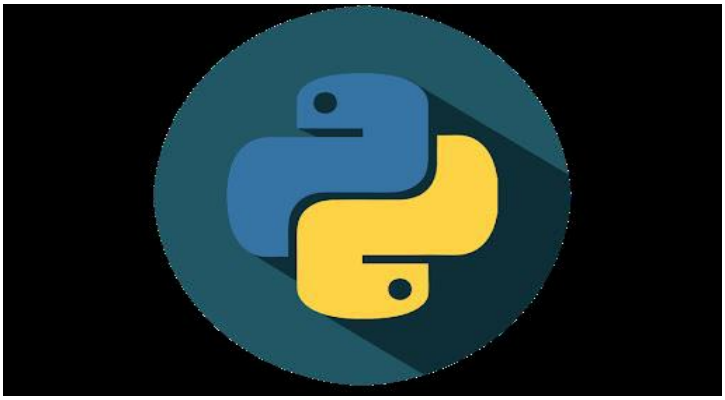

Fig 3.12 Python

Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It is designed to be easy to read, write, and understand, and is widely used in many fields such as data science, web development, scientific computing, and more.
Python has a simple syntax and is dynamically typed, which means that you don't have to declare the data type of a variable before using it. This makes Python code more concise and easier to read compared to other programming languages.

Python also has a large standard library that provides a wide range of functionality, including file I/O, networking, regular expressions, and more. There are also many third-party libraries available that can be easily installed using tools such as pip, which is the default package installer for Python.

Python can be used for a variety of tasks, including scripting, automation, web development, scientific computing, machine learning, and more. It has become one of the most popular programming languages in the world, and its popularity continues to grow as more developers and organizations adopt it for their projects.

Fig 3.13 SkLearn

scikit-learn, commonly abbreviated as sklearn, is a popular Python library for machine learning. It provides a range of algorithms for various machine learning tasks, including classification, regression, clustering, and dimensionality reduction, as well as tools for preprocessing data, model selection, and evaluation.

In addition to the algorithms, scikit-learn provides a range of utilities for data preprocessing and model selection, such as:

Data preprocessing: Scaling, centering, normalization, feature selection, and more
Cross-validation: splitting data into training and testing sets, k-fold cross-validation, and more
Model selection: Grid search, randomized search, model evaluation metrics, and more

Fig 3.14 TensorFlow

TensorFlow is an open-source machine learning library developed by Google that is widely used in industry and academia. It provides a flexible and efficient platform for building and training machine learning models.

TensorFlow includes several key features, including:

1. Automatic differentiation: TensorFlow can automatically compute gradients for a wide range of mathematical operations, which is essential for training deep neural networks.
2. Dataflow graphs: TensorFlow models are defined as directed graphs, where nodes represent mathematical operations and edges represent the data that flows between them. This provides a flexible and efficient way to represent complex computations.
3. Hardware acceleration: TensorFlow can take advantage of GPUs and other specialized hardware to accelerate model training and inference.
4. High-level APIs: TensorFlow includes high-level APIs, such as Keras, that make it easy to define and train deep neural networks.

TensorFlow supports a wide range of machine learning tasks, including classification, regression, clustering, and more. It is widely used in industry and academia for building and training deep neural networks for tasks such as image recognition, natural language processing, and more.

Fig 3.15 Keras

Keras is a high-level open-source neural network library written in Python that is built on top of TensorFlow. It provides a simple and user-friendly interface for building and training deep neural networks, allowing developers to quickly prototype and experiment with different architectures and hyperparameters.

Keras provides a range of pre-built layers that can be used to construct neural networks, including convolutional layers, recurrent layers, dense layers, and more. It also includes a range of activation functions, loss functions, and optimizers that can be used to train models.

One of the key features of Keras is its simplicity and ease of use. It allows developers to define neural network models using a simple and intuitive API, and provides powerful tools for monitoring and visualizing training progress.

Keras is also highly modular and flexible, allowing developers to easily create custom layers and loss functions, and to integrate with other libraries and tools.

Keras supports a range of use cases, including image classification, object detection, natural language processing, and more. It has been widely adopted in industry and academia, and is often used in conjunction with other machine learning libraries such as TensorFlow and scikit-learn.

Fig 3.16 OpenCV

OpenCV (Open Source Computer Vision) is an open-source computer vision and machine learning library that is designed to provide a common infrastructure for computer vision applications. It is written in C++ and Python and is widely used in industry and academia.

OpenCV provides a range of computer vision algorithms and tools, including:

- Image and video processing: Loading, displaying, and saving images and videos, as well as tools for image and video filtering, segmentation, and feature extraction.
- Object detection: Tools for detecting objects in images and videos, including face detection, pedestrian detection, and more.
- Machine learning: OpenCV provides a range of machine learning algorithms, including decision trees, random forests, support vector machines, and more.
- Camera calibration: Tools for calibrating cameras, including intrinsic and extrinsic camera parameters.
- 3D reconstruction: Tools for reconstructing 3D scenes from multiple images or video streams.

OpenCV is highly optimized for performance and can take advantage of multi-core processors and GPUs to accelerate computation. It is cross-platform and can run on a wide range of operating systems, including Windows, Linux, macOS, iOS, and Android.

OpenCV is widely used in a range of applications, including robotics, augmented reality, surveillance, automotive, and more.

3.6 Source code:

```
import os       #imported some necessary libraries.
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
import cv2
from tensorflow.keras.preprocessing.image import ImageDataGenerator , load_img
,img_to_array
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, Conv2D,
MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report,confusion_matrix


my_data_dir =  'C:/Users/ASUS/Desktop/label/'  #created a dataset.
train_path = my_data_dir + 'train/'       #training path.
test_path = my_data_dir + 'test/'          #testing path.

image_shape = (500,217,1)
batch_size = 16


img = plt.imread('C:/Users/ASUS/Desktop/label/train/ok/def_0_341.jpeg')
plt.figure(figsize=(12,8))
plt.imshow(img,cmap='gray')    #reading a image.


image_gen = ImageDataGenerator(rescale=1/255)
train_set=image_gen.flow_from_directory(train_path,target_size=image_shape[:2],color_mo
de="grayscale",batch_size=batch_size,class_mode='binary',shuffle=True)

test_set=image_gen.flow_from_directory(test_path,target_size=image_shape[:2],color_mode
="grayscale",batch_size=batch_size,class_mode='binary',shuffle=False)

train_set.class_indices    #binary classification system

model = Sequential()  #CNN model
```

```
model.add(Conv2D(filters=8, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=16, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=16, kernel_size=(3,3),input_shape=image_shape,
activation='relu',))
model.add(MaxPooling2D(pool_size=(2, 2)))




model.add(Flatten())

model.add(Dense(224))
model.add(Activation('relu'))

# Last layer
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
early_stop = EarlyStopping(monitor='val_loss',patience=2)

results = model.fit_generator(train_set,epochs=20,
                              validation_data=test_set,
                             callbacks=[early_stop])

pred_probability = model.predict(test_set)

predictions = pred_probability > 0.5

print(classification_report(test_set.classes,predictions))

plt.figure(figsize=(10,6))
sns.heatmap(confusion_matrix(test_set.classes,predictions),annot=True)
img = cv2.imread(train_path+'ok/def_0_635.jpeg',0)
img = img/255 #rescalinng
pred_img =img.copy()
plt.figure(figsize=(12,8))
plt.imshow(img)

prediction = model.predict(img.reshape(-1,500,217,1))
print(prediction)
if (prediction<0.5):
```

```
    print("def_front")
    cv2.putText(pred_img, "def_front", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,
255, 0), 2)
else:
    print("ok_front")
    cv2.putText(pred_img, "ok_front", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,
255, 0), 2)

plt.imshow(pred_img,cmap='gray')
plt.axis('off')
plt.show()
img1 = cv2.imread(test_path+'def/def_0_438.jpeg',0)
img1 = img1/255
pred_img1 =img1.copy()
plt.figure(figsize=(12,8))
plt.imshow(img1,cmap='gray')

prediction = model.predict(img1.reshape(-1,500,217,1))
if (prediction<0.5):
    print("def_front")
    cv2.putText(pred_img1, "def_front", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,
255, 0), 2)
else:
    print("ok_front")
    cv2.putText(pred_img1, "ok_front", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0,
255, 0), 2)

plt.imshow(pred_img1,cmap='gray')
plt.axis('off')
plt.show()

prediction = model.predict(img1.reshape(-1,500,217,1))


print(prediction)


import cv2
from tensorflow import keras

model = keras.models.load_model("label_inspection.h5")
cap = cv2.VideoCapture(1)

while True:
    ret, frame = cap.read()
    if ret:


        cv2.rectangle(frame, (200, 0), (420, 500), (255, 255, 255), 3)
        # rectangle for computer to play
```

```
        # extract the region of image within the user rectangle
        roi = frame[0:500, 200:420]
        img = cv2.cvtColor(roi, cv2.COLOR_RGB2GRAY)
        img = cv2.resize(img, (217, 500))
        img1 = img
        img1 = img1/255
        prediction = model.predict(img1.reshape(-1,500,217,1))
        print(prediction)


        font = cv2.FONT_HERSHEY_SIMPLEX
        if (prediction>0.5):
            print("ok")

            cv2.putText(frame, "OK",(50, 50), font, 1.2, (0, 255, 0), 2, cv2.LINE_AA)


        else:
            print("defective")
            cv2.putText(frame, "Defective",(50, 50), font, 1.2, (0, 0, 255), 2, cv2.LINE_AA)


        cv2.imshow('Color Frame', frame)
        key = cv2.waitKey(50)
        if key == ord('q'):
            break

    else:
        print('Frame not available')
        print(cap.isOpened())

    # rectangle for user to play



cap.release()
cv2.destroyAllWindows()
```

3.7 Simulation:

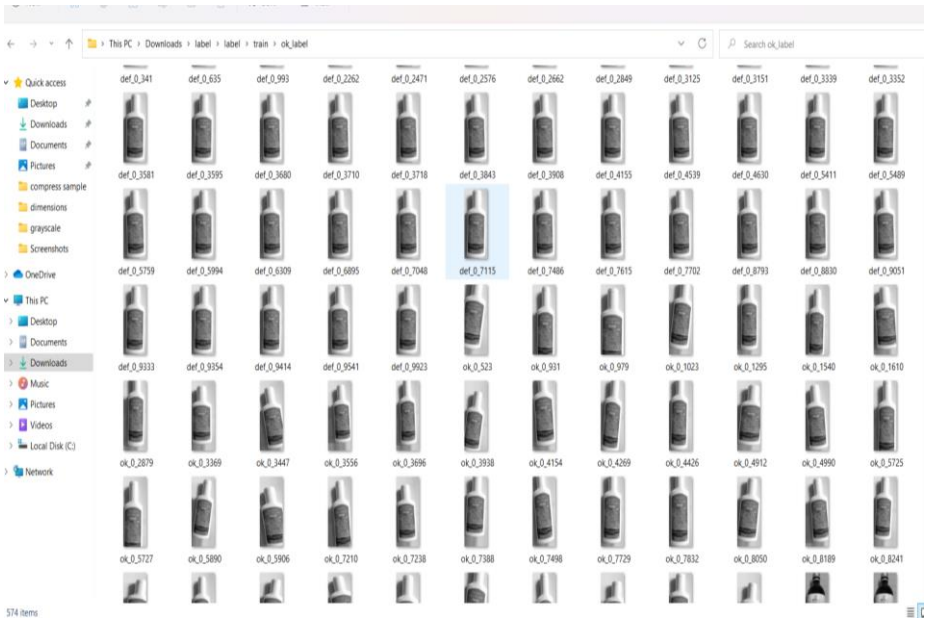

Fig 3.17 Dataset Directory Where We Have Stored Labeled Dataset.

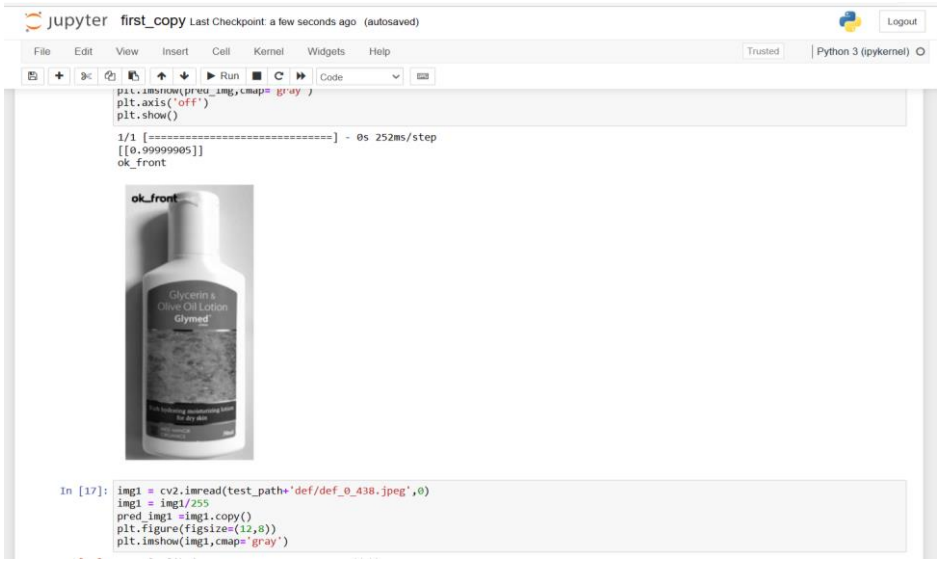

Fig 3.18 Detecting Proper Product.

In above figure, we have given a image to CNN model and predicted the result. Model is predicting this as a ok product.

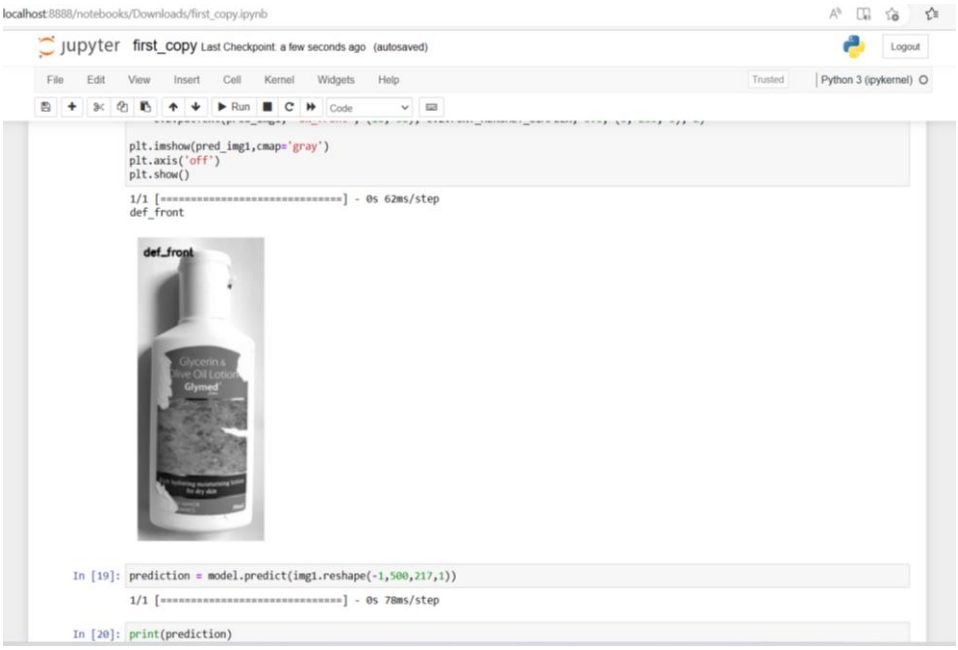

Fig 3.19 Detecting Defective Product.

In this above figure, label of this product is tear and model is predicting this as a defective product.

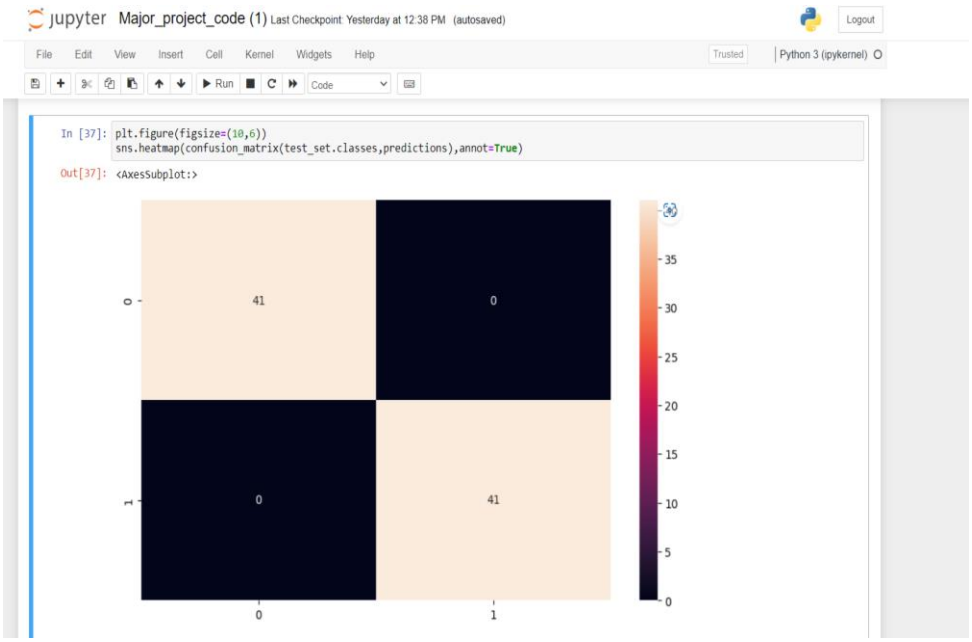

Fig 3.20 Classification Report

In this above figure, top white box has showing 41 images are true positive it means that these images are predicted true and actually these are true. basically proper images are showing proper and defective images are showing defective as well.

	precision	recall	f1 -score	support
0	1.00	0.99	0.99	711
1	0.99	1.00	0.99	684
accuracy			0.99	1395
Macro avg.	0.99	0.99	0.99	1395
Weighted avg.	0.99	0.99	0.99	1395

Table 3.1 Classification Report

Precision =Formula for Precision=$\frac{TP}{TP+FP}$, where TP=True Positive

FP=False Positive.

Precision measures the accuracy of positive predictions, or how often the algorithm correctly predicts a positive outcome. A high precision value indicates that the algorithm is good at avoiding false positives.

Precision is useful in situations where the cost of a false positive prediction is high, such as in medical diagnosis or fraud detection. For example, in a medical test, a false positive result could lead to unnecessary treatment or surgery, so a high precision value is desirable to minimize the occurrence of false positives.

Recall $=\frac{TP}{TP+FN}$, where FN=False Negative

Recall measures the ability of the algorithm to correctly identify positive cases, or how often the algorithm correctly predicts a positive outcome out of all positive cases in the data. A high recall value indicates that the algorithm is good at avoiding false negatives.

F1-Score $= 2 * \frac{(precision*recall)}{precision+recall}$

The F1 score is a performance metric that is calculated as the harmonic mean of precision and recall. It provides a single measure of the overall performance of a classification algorithm, taking into account both precision and recall.

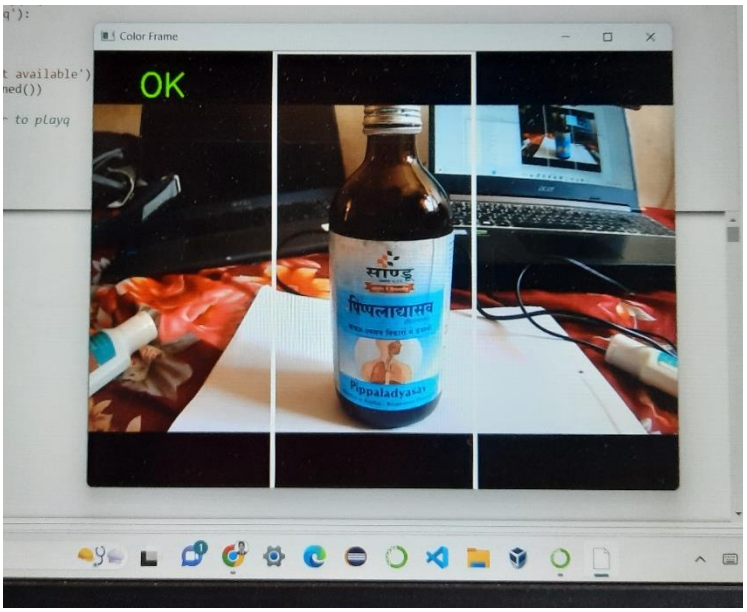

Fig 3.21 System is recognizing as ok product.

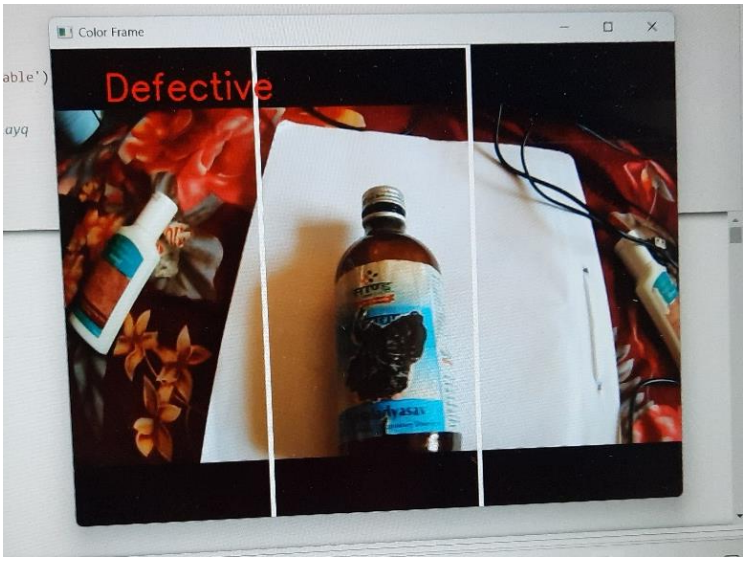

Fig 3.22 System is Recognizing as Defective Product.

Chapter 4

Conclusion and Result

4.1Result

Fig 4.1 Ok Front Image.

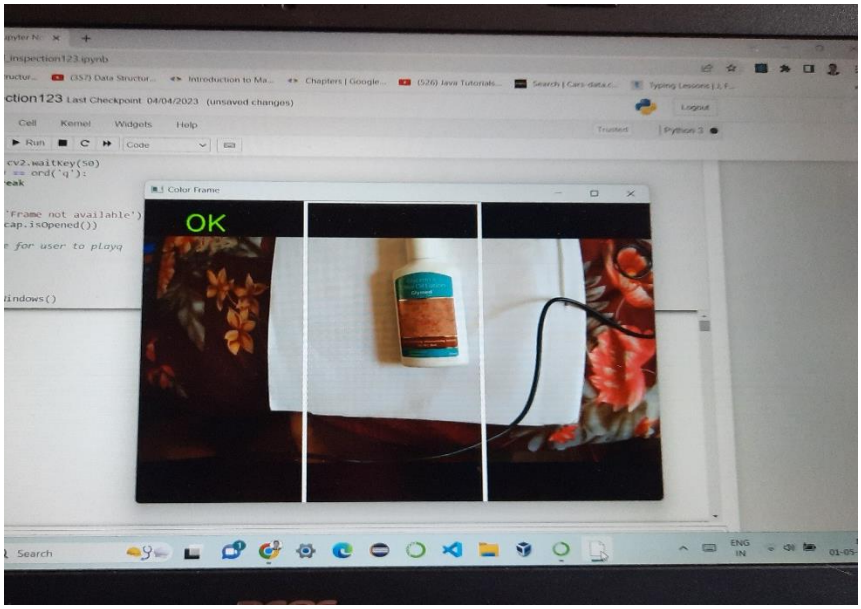

Fig 4.2 Prediction for Ok Product.

Fig 4.3 Defective Front Image.

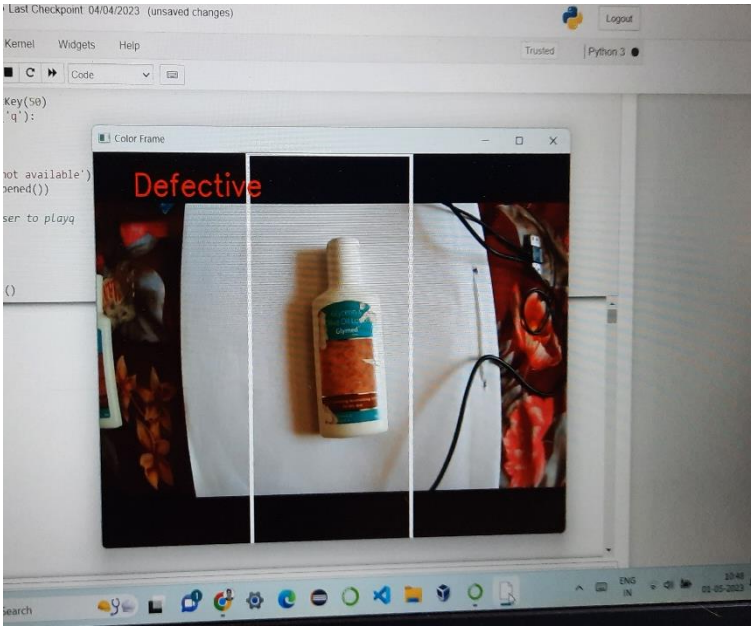

Fig 4.4 Prediction for Defective Product.

4.2 Future Scope

1.Food & Pack Labelling:

Food and packaging labels are becoming more complex, with variable data, traceability devices and RFID security tags. Demands for increased throughput, improved process efficiency, and quality assurance are driving the dependency on inspection to reduce the risk of costly product returns, enhanced client satisfaction, and ultimately protect profitability. Our Discovery systems focus on print and label inspection requirements specifically for the food and packaging environment. Return on investment can be quickly achieved through reduced inspection costs, improved efficiency, waste reduction, and reduced risk.

2.Pharma, Shipping & Security Labels:

Discovery's standard verification, quality control and reporting tools can easily be applied to pharmaceutical, shipping and security labels such as tax stamps and authentication/anti-counterfeit labels. Our solutions read critical data even from challenging surfaces, and UV/IR printed codes and validates them for accuracy, integrity (missing/duplication detection) and compliance to industry-standard protocols. Our Discovery suite can easily communicate with an MIS system associated with a pharmaceutical track & trace or lifecycle tracking system within a converting or packaging facility.

4.3 Conclusion

In conclusion, a label inspection system is an essential quality control tool that can help manufacturers ensure the accuracy and consistency of labels on their products. With advancements in machine vision and deep learning technologies, the accuracy and efficiency of label inspection systems can be significantly improved.

Convolutional Neural Networks (CNNs) have shown great promise in enhancing the accuracy and efficiency of label inspection systems. They can learn complex features of images, classify different types of label defects, and provide real-time feedback to operators and supervisors. Further advancements in CNN architectures, transfer learning, and object detection can further enhance the performance of label inspection systems.

Overall, label inspection systems using CNNs have the potential to transform the manufacturing industry by improving product quality and reducing costs associated with faulty products.

References

[1]S. Suresh and S. N. Omkar,"A comprehensive review on automated visual inspection systems for quality control of products" ,2018.
[2]S.. Zhao et al,"A deep learning-based approach for label inspection in pharmaceutical industry" ,2020.
[3]A. V. Bongale and A. G. Keskar ,"A deep learning-based approach for label inspection in pharmaceutical industry",2020.
[4]H. Zheng et al,"Label detection and recognition in logistics using deep learning" ,2019.
[5]Traian Rebedea. “Expiry date recognition using deep neural networks,”, January 2020.
[6] N. Mohd Saad, Nik Mohd Zarifie Hashim. “Barcode Recognition System.” July 2013 International Journal of Emerging Trends & Technology in Computer Science 2(4):278-283
[7]Lokesha Hlmg, Shilpashree K.S. “Implementation of Image Processing on Raspberry Pi.” May 2015 IJARCCE 4(5):199-202
[8] T. Ananthan,Puranam Revanth Kumar,” Machine Vision using LabVIEW for Label Inspection.” Conference: Journal of Innovation in Computer Science and Engineering At: Hyderabad, Telangana, India Volume: 9, Issue.1, pp. 58-62
[9] Evgenicy krasnakutsky.” AI Visual Inspection for defect detection ”,Aug 2021.
[10] Xiaokang Ren.” Research And Application of Label Defect Detection Method Base On Machine Learning”, Jan 2020.
[11] Ch Anush.“Bottle Line Detection using Digital Image Processing With Machine learning”, Aug 2021.
[12] Anand Paul.” Image Processing for quality Management in Industrial IOT ”,Oct 2019.
[13] Changsheng Li.” Supervision Machine vision detection method based on AI defect simulation ”,Nov 2020.
[14] Tao Yu.” Label defect detection based on Machine vision ”,March 2022.
[15] Rahmatov Nematullo.” Machine Learning-based automated image processing ”,Oct 2019.
[16] Mehshan Khan etal.” A smart and robust Automatic Inspection of Printed labels”,March 2022.
[17] Rohit Bhisey.” Packaging and label inspection Machine May see a big move”,Aug 2018.
[18] Yu Wu1 and Yanjie Lu2.” Machine Vision System for detecting surface defects on packing boxes based on support vector”,May 2019