FSM Online Internship Completion

Report on

[INTP23-ML-9 Piston Defect Detection Using Computer Vision]

In

[Machine Learning]

Submitted By
[Nandan Vaid]
{Galgotias University}

Under Mentorship of
[Mr Keivalya Pandya]

IITD-AIA Foundation for Smart Manufacturing

[1-June-2023 to 31-July-2023]
[Piston Defect Detection Using Computer Vision]

Abstract

As technology advances, making high-quality engine pistons has become really important. But sometimes, there are tiny flaws on the piston surface that are hard to spot using normal methods. Right now, most companies use methods where they touch or look at the piston manually. However, these methods often miss the small flaws because the piston's surface is shiny and tricky to check.

So, a smart solution using machines and cameras has been created. It looks at the piston closely and can find these small flaws easily. It uses special steps like preparing the picture, making it black and white, fixing its shape, and setting a limit. This new system is trustworthy, works fast, and is accurate. It can help in making pistons better and can be used in real factories to improve the pistons we use in cars and machines

Keywords: Piston ,Visualization ,Machine learning, Computer Vision ,Data preprocessing.

Table of Content

1.Introduction

This project marks a significant stride forward – Piston anomaly recognition .The profound importance of impeccable piston surfaces resonates across the realms of performance and safety, underscoring the imperative for precise defect identification.

Traditionally, the task of identifying such imperfections has been entrusted to manual inspection – a process fraught with subjectivity and inefficiency. However, a convergence of cutting-edge technologies has ignited a paradigm shift – the advent of machine vision. Through the orchestration of high-resolution cameras, intricate image analysis, and data-driven decision-making, this technology stands on the brink of redefining defect detection, promising unmatched precision and celerity.

Within the pages of this project, we embark on an odyssey that unravels the intricacies of machine vision's application in the realm of piston surface defect detection. From capturing intricate images with meticulous detail to the strategic deployment of advanced algorithms for the discernment of defects, we traverse every contour of this transformative journey. Moreover, we chart the course of its seamless assimilation into manufacturing processes, heralding an era of real-time quality assurance and the optimization of processes. By illuminating the symbiotic alliance between innovation and precision, this research not only enriches our project's narrative but also contributes substantively to the overarching dialogue on technological progress in the realm of manufacturing quality control.

2.Problem Statement

- In the world of making pistons, it's super important to have pistons with perfect surfaces. Even small problems can make the pistons work badly or even be unsafe. Right now, people check pistons for problems, but this way is slow, not very accurate, and can cost a lot.

- Our project is all about fixing this. We want to use computers and special cameras to create a smart Piston Defect Detection system. This system will take pictures of pistons and use clever math to find any issues on the surface, like scratches or cracks. The cool thing is, it'll do this really fast and always the same way, so it's super accurate.

- We're working on making this system smart enough to fit right into the way pistons are made. The main goals are: 1) Making the computer system that takes great pictures of pistons. 2) Teaching the computer how to find defects on the pictures using smart tricks. 3) Putting this system into the manufacturing process so it can help make good pistons all the time. 4) Testing the system to see if it's better than people at finding problems.

- By doing all this, we want to make making pistons much better and easier, saving time, money, and making sure they work great.

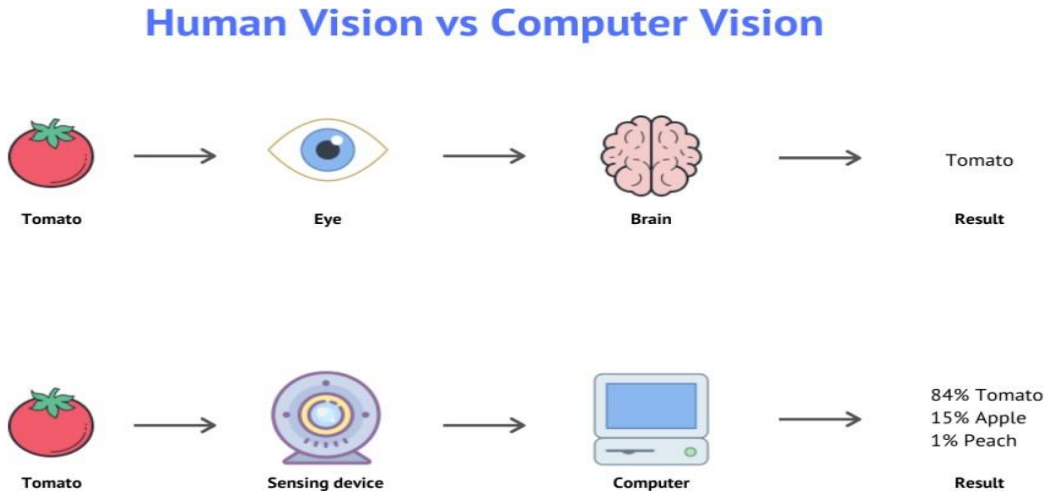
3.Existing Solutions

In this Piston Defect Detection project , existing solutions predominantly involve contact-based methods and manual inspection. These approaches, while partially effective, struggle with accurately identifying small and irregular surface defects on engine pistons. Traditional machine vision techniques, including basic image processing, offer limited accuracy and efficiency in tackling the complexities of piston surfaces. Recent advancements in deep learning have explored automated defect identification, but their success relies heavily on training data quality.

In our project, we aimed to enhance defect detection by harnessing advanced machine vision and tailored image processing within the HALCON framework.

Our approach sought to overcome limitations of traditional methods, ensuring improved accuracy, defect visibility, and operational efficiency in practical manufacturing scenarios.

By leveraging the strengths of both machine vision and specialized image processing, we aimed to contribute a novel solution to the challenge of Piston Defect Detection using Computer Vision.



4.Proposed Development

Process and Issue Description

The purpose of the inspection machine is to detect a certain class of defects, to sort the engine blocks on the production line, and to wash the bottom part of the block using a special system for removing dirt, dust, and other mechanical process impurities .When a defect is detected, the engine block is removed trough a conveyor from the production line. In the washing process, some special solvents and emulsions are used .The implementation completes the already installed inspection machine by adding a new station with the purpose to automate the visual inspection performed until now by the operator**.**

The complete layout of the process can be observed in Figure 1.

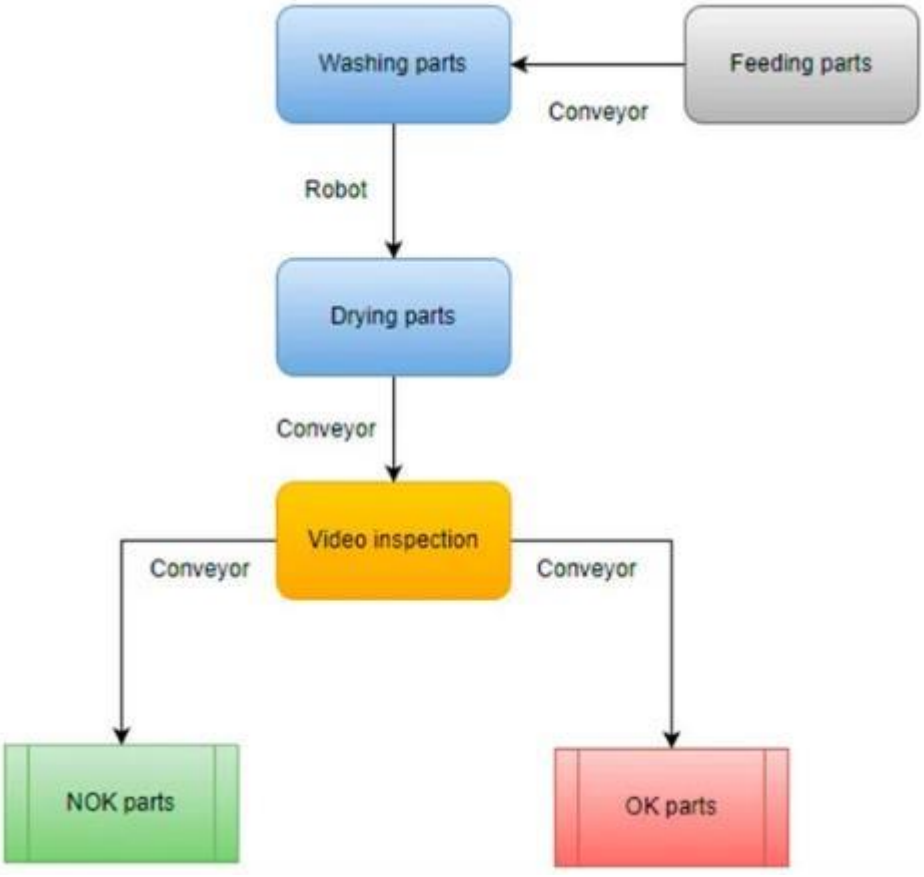


Figure 1. Architecture of the washing and sorting machine on the production line.

The next step after washing will be the drying and cleaning of the block with an airflow through the bottom part and the cylinder chambers. The drying process leaves behind dried cleaning emulsion, which will make the automated inspection more difficult.

The engine block is made from cast iron with green sand insertions The damaged sand core can generate inclusions inside or at the surface of the part. Another defect is generated by the impossibility of removing all the gases from the mold when the liquid metal takes contact with the surface of the mold. This process involves generating blowholes.

4.2 Architecture Description

For moving the camera, a PLC that is controlled directly by the main inspection machine was used. When the PLC receives the trigger, a stepper motor is actuated. The camera is hovered over each of the cylinders for acquisition and is connected to the stepper motor with a coupling in order to create a linear movement. When the camera is in position, the acquisition and processing system is triggered.
Figure 2 describes the system architecture including the sensor.

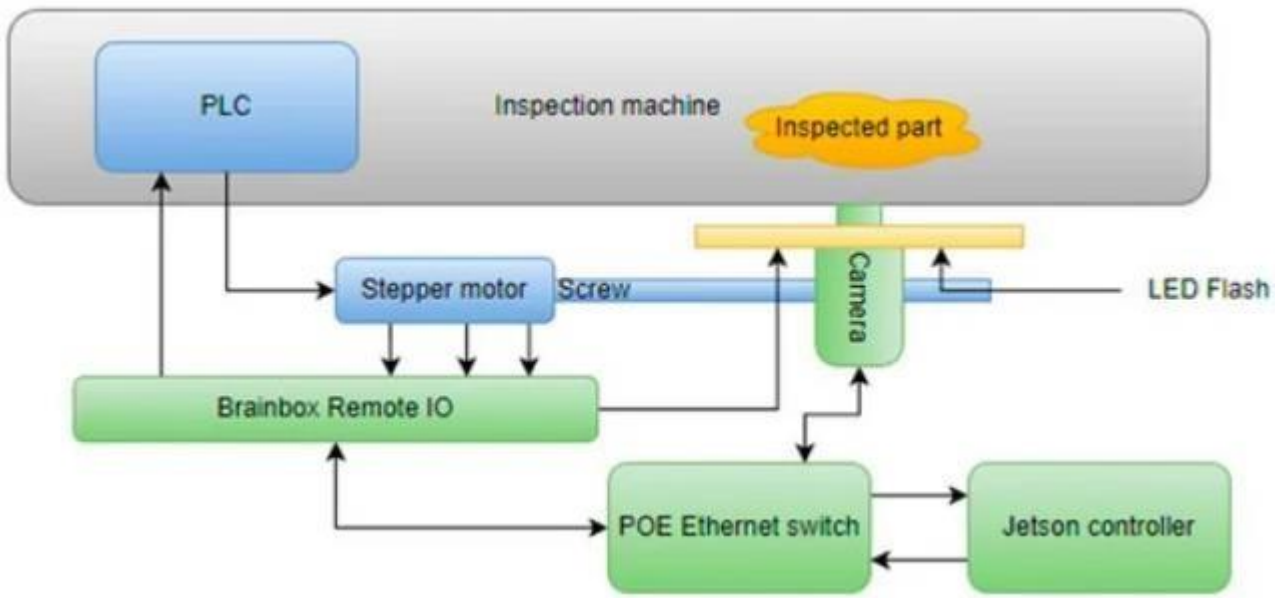


Figure 2 . System architecture of the video inspection operation

4.3 Hardware Description

For implementing the computer vision solution, the following hardware components were used:

- Nvidia Jetson Nano controller with 4 GB RAM memory, 128 core GPU, A57 Quad Core CPU
- The Imaging Source DMK 33GX290e Monochrome Camera
- ED-008 Ethernet to Digital I/O Brainboxes module
- EffiLux LED Infrared Light flash EFFI-FD
- Industrial compliant POE Ethernet Switch

4.4 Software Description

The application was implemented using Python programming language.

Software was designed to cover all of the manufacturing necessities, e.g., logging, user management, process handling.

Figure 3 , a complete sequence diagram of the process can be observed.

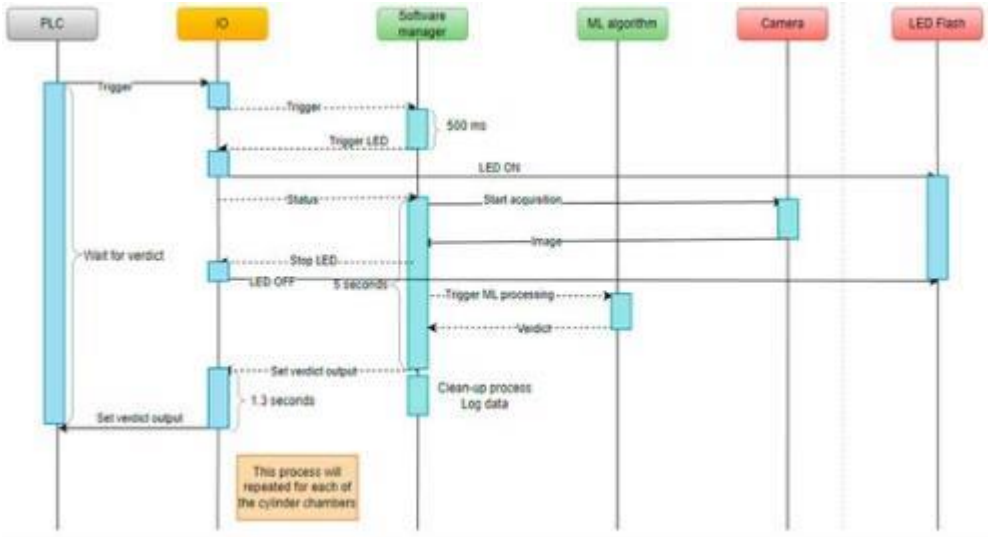


Figure 3 . Process sequence diagram

4.5 Processing Algorithm Description

The processing algorithm has two main parts: the conventional processing and the prediction using a machine learning model. The conventional processing works

very well when there are no significant traces of cleaning emulsion on the cylinder. When the emulsion becomes mixed with dust, traces become increasingly noticeable and with a lower grayscale level. The second part of the processing algorithm is the convolutional neural network implemented using the PyTorch framework. The first layer takes as input the three RGB channels of the image and splits it in eight features for the next layer. The second layer is a max pooling layer with a kernel size of 3 × 3 and with padding enabled for looping through the entire image.Third layer is another convolutional layer, similar to the first layer.

5. Functional Implementation

Cross-checking every dataset, preprocessing and visualization:

[ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2
import random
%matplotlib inline
```

[ ]:

```
base_path = '../input/Nandan\Desktop\fsm\dataset\perfect'
```

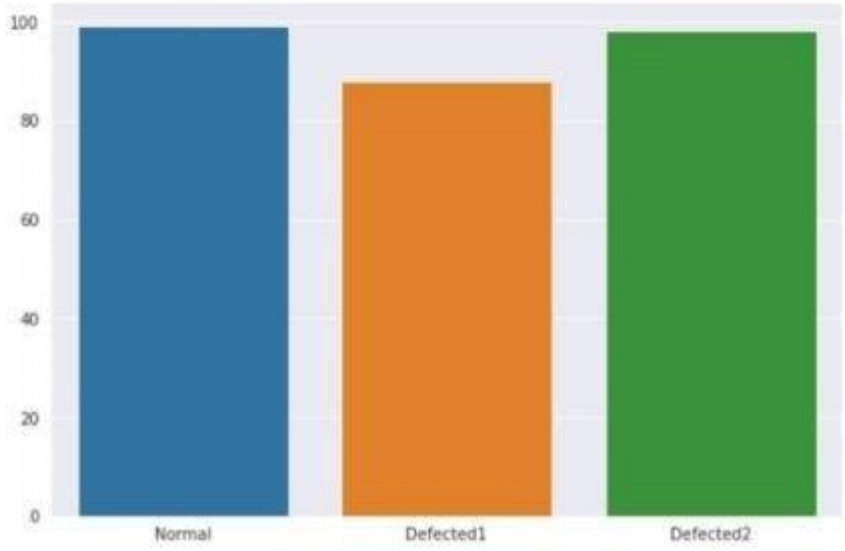
Cross-checking every dataset, what is the count of images in each dataset?

In [28]:
```
print(f"There are {len(os.listdir(base_path))} type of dataset")
```

```
There are 3 type of dataset
```

In [29]:
```
data_set = []
amount_of_each = []
for folder in os.listdir(base_path):
    data_set.append(folder)
    amount_of_each.append(len(os.listdir(os.path.join(base_path,fold
er))))

plt.figure(figsize=(9,6))
sns.set_style('darkgrid')
sns.barplot(x = data_set,y = amount_of_each)
```

Out[29]:
```
<AxesSubplot:>
```

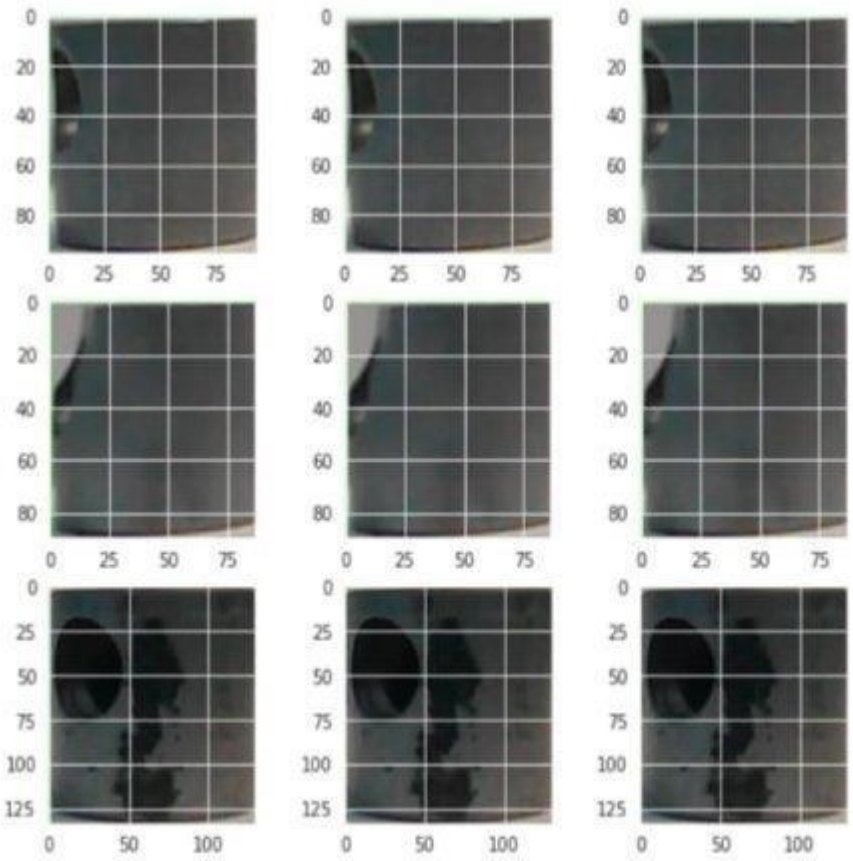


Training Accuracy – 0.99978856869567
Testing Accuracy - 0.9984156869032

Visualize each type of dataset

In [30]:

```
fig,ax = plt.subplots(3,3,figsize=(9,8))
for i in range(3):
    file_name = random.choice(os.listdir(os.path.join(base_path,data
_set[i])))
    folder = os.path.join(base_path,data_set[i])
    for j in range(3):
        img = cv2.imread(os.path.join(folder,file_name))
        ax[i,j].imshow(img)
```



In [31]:
```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,Flatten,Dropout
from tensorflow.keras.optimizers import RMSprop,Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16,ResNet50,EfficientNe
tB6
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint
import tensorflow as tf
tf.config.list_physical_devices('GPU')
```

Out[31]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

Created model by using Resnet pretrained model

In [32]:
```
resnet = ResNet50(include_top=False,input_shape=(80,80,3))
for layer in resnet.layers:
    layer.trainable = False
flat = Flatten()((resnet.layers[-1].output))
dense1 = Dense(1024,activation='relu')(flat)
dense2 = Dense(512,activation='relu')(dense1)
drop = Dropout(0.2)(dense2)
model_output = Dense(3,activation = 'softmax')(drop)
model = Model(resnet.input,model_output)
# model.summary()
```

In [33]:
```
callback = ModelCheckpoint('./checkpoint.ckt',save_weights_only=Tru
e,
    monitor='val_accuracy',
    mode='max',
    save_best_only=True)
earlystop = EarlyStopping(monitor='val_accuracy',patience=15,mode='m
ax',restore_best_weights=True)
model.compile(optimizer=RMSprop(learning_rate=0.001),loss = 'categor
ical_crossentropy',metrics=['accuracy'])
```

In [34]:
```
data_gen = ImageDataGenerator(brightness_range=[1.5,2.5],
                              rotation_range = 0.6,validation_split=
0.3)
```

In [35]:
```
train_data = data_gen.flow_from_directory(base_path,target_size=(80,
80),subset='training')
valid_data = data_gen.flow_from_directory(base_path,target_size=(80,
80),subset='validation')
```

```
Found 201 images belonging to 3 classes.
Found 84 images belonging to 3 classes.
```

In [36]:

```
history= model.fit(train_data,epochs=8,validation_data=valid_data,ca
llbacks=[callback])
```

```
Epoch 1/8
7/7 [==============================] - 5s 311ms/step - loss: 23.99
00 - accuracy: 0.4112 - val_loss: 0.4788 - val_accuracy: 0.9286
Epoch 2/8
7/7 [==============================] - 1s 145ms/step - loss: 0.761
6 - accuracy: 0.9324 - val_loss: 0.2620 - val_accuracy: 0.9405
Epoch 3/8
7/7 [==============================] - 1s 172ms/step - loss: 0.011
9 - accuracy: 0.9930 - val_loss: 0.8194 - val_accuracy: 0.8452
Epoch 4/8
7/7 [==============================] - 1s 160ms/step - loss: 0.001
6 - accuracy: 1.0000 - val_loss: 0.8141 - val_accuracy: 0.8810
Epoch 5/8
7/7 [==============================] - 1s 154ms/step - loss: 0.034
1 - accuracy: 0.9944 - val_loss: 7.2809 - val_accuracy: 0.6786
Epoch 6/8
7/7 [==============================] - 1s 160ms/step - loss: 2.602
1 - accuracy: 0.9051 - val_loss: 0.5025 - val_accuracy: 0.9167
Epoch 7/8
7/7 [==============================] - 1s 147ms/step - loss: 0.013
2 - accuracy: 0.9988 - val_loss: 2.5682 - val_accuracy: 0.8571
Epoch 8/8
7/7 [==============================] - 1s 143ms/step - loss: 0.082
5 - accuracy: 0.9683 - val_loss: 2.8171 - val_accuracy: 0.8571
```

6.Final Deliverable

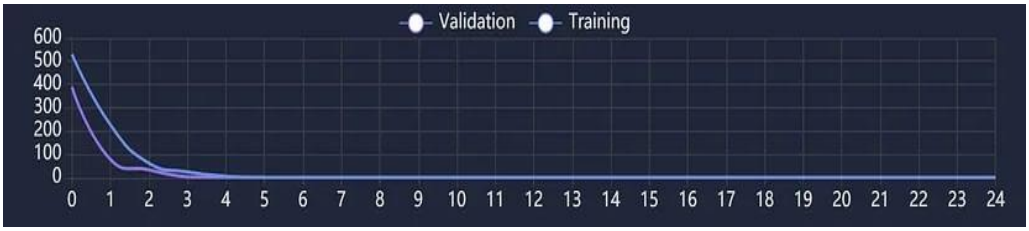
For deployment purpose Keras is used .

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

# Load and preprocess the sample image
sample_image_path = '/content/drive/MyDrive/xyz/Normal/kumda_componentN10.png'
img = image.load_img(sample_image_path, target_size=(80, 80))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
preprocessed_img = img_array / 255.0  # Normalize the pixel values

# Make predictions
predictions = model.predict(preprocessed_img)

# Interpret the prediction results
class_labels = ['defactive1', 'defactive2', 'normal']
predicted_class_index = np.argmax(predictions)
predicted_class_label = class_labels[predicted_class_index]
confidence = predictions[0][predicted_class_index]

print("Predicted Class:", predicted_class_label)
print("Confidence:", confidence)
```

```
1/1 [==============================] - 0s 149ms/step
Predicted Class: normal
Confidence: 0.5174713
```

Accuracy plot:

Loss Plot:



7.Scalability to Solve Industrial Problem

Scalability proves indispensable in resolving industrial dilemmas, as demonstrated in its application to Piston Defect Detection via Computer Vision. By leveraging scalable solutions, manufacturers can efficiently analyze and process large volumes of visual data from production lines, enabling the timely and accurate identification of piston defects .

The ability to seamlessly expand the system's capacity ensures consistent performance as production scales up, maintaining high precision in defect recognition. This not only enhances product quality but also optimizes resource utilization and reduces manual intervention, leading to increased efficiency and reduced costs. Thus, scalability stands as a cornerstone in effectively tackling industrial issues, such as piston defect detection, through advanced technologies like computer vision.

By implementing computer vision, businesses are more likely to optimize their processes and achieve their scaling goals. As the computer vision tool resolves the bottleneck of human quality control inspection, the manufacturers can build in other areas.

References

1) https://www.freecodecamp.org/
2) Li, J.; Su, Z.; Geng, J.; Yin, Y. Real-time Detection of Steel Strip Surface Defects Based on Improved YOLO Detection Network. *IFAC-Pap.* **2018**, *51*, 76–81. [Google Scholar] [CrossRef]
3) Hoda, R.; Salleh, N.; Grundy, J. The Rise and Evolution of Agile Software Development. *IEEE Softw.* **2018**, *35*, 58–63. [Google Scholar] [CrossRef]
4) Tao, X.; Zhang, D.; Ma, W.; Liu, X.; Xu, D. Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks. *Appl. Sci.* **2018**, *8*, 1575. [Google Scholar] [CrossRef][Green Version]
5) https://ieeexplore.ieee.org/document/9734579/