INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)

www.ijprems.com
editor@ijprems.com

Vol. 04, Issue 05, May 2024, pp: 2036-2046

e-ISSN : 2583-1062

Impact Factor: 5.725

SOLAR CELL DEFECT DETECTION BASED ON OPTIMIZED YOLOV5

Dr Ch. Vijayakumar[1], Changalwala Kameswara Vishnu Vallabha[2], Chebrolu Bangaru Sai Prasanth[3], Potlacheruvu Sanjay[4], Palla Rahul[5]

[1]Associate Professor, CSE Dept, ACE Engineering College, Hyderabad, India.

[2,3,4,5]Student, CSE Dept, ACE Engineering College, Hyderabad, India.

ABSTRACT

Traditional vision methods for solar cell defect detection have problems such as low accuracy and few types of detection, so this paper proposes an optimized YOLOv5 model for more accurate and comprehensive identification of defects in solar cells. The model firstly integrates five data enhancement methods, namely Mosaic, Mixup, hsv transform, scale transform and flip, to expand the existing data set to improve the feature training accuracy and enhance the robustness of the model; secondly, CA attention mechanism is introduced to improve the feature extraction ability of the model; to address the problems of different target defect classification and localization concerns, the detection head in the original model is replaced with a decoupling head, which significantly improve the detection accuracy of the model without affecting the convergence speed of the model. The results show that the optimized model achieves an mAP of 96.1% on the publicly available dichotomous ELPV dataset, and can identify and locate a variety of common defects in the PVEL-AD dataset, while the mAP can reach 87.4%, an improvement of 10.38% compared with the original YOLOv5 model, which enables the model to perform more accurately while ensuring the real-time requirement of solar cell surface defects detection task.

1. INTRODUCTION

In the quest for sustainable and renewable energy, solar power has emerged as a key player, with solar cells at the core of this technology, converting sunlight into electricity. However, the efficiency and longevity of solar cells are often compromised by surface defects, such as cracks, scratches, or contamination, which can significantly reduce performance and lead to premature failure. Early and accurate detection of these defects is crucial to ensure optimal functionality and reduce manufacturing losses. Traditional methods of defect detection are often manual, labor-intensive, and prone to human error, highlighting the need for automated solutions. Leveraging advancements in machine learning, particularly convolutional neural networks (CNNs), offers a promising avenue for this challenge. The YOLO (You Only Look Once) model, known for its speed and accuracy in real-time object detection, presents a viable solution for automated defect detection. This project focuses on optimizing the YOLOv5 model to enhance its capability in identifying and classifying surface defects in solar cells, aiming to improve detection accuracy and operational efficiency in industrial applications. By incorporating novel enhancements such as hybrid attention mechanisms, advanced data augmentation techniques, and adaptive anchor box strategies, our approach aims to set a new benchmark in the field of solar cell defect detection...

2. OBJECTIVES

In our project there are 4 objectives. They can be listed as:

- Detection
- Optimization
- Accuracy
- Automation

3. METHODOLOGY

The methodology for this project includes collecting and preprocessing a dataset of solar cell images, applying advanced data augmentation techniques like Mosaic, MixUp, and GANs for synthetic defect generation. The YOLOv5 model is enhanced with deformable convolution layers and a hybrid attention mechanism combining Efficient Channel Attention (ECA) with spatial attention modules.

An adaptive anchor box optimization using a modified K-means++ algorithm and a composite loss function integrating CIOU with an overlap penalty term are implemented. The model's performance is then evaluated using metrics such as mean Average Precision (mAP), recall, precision, and inference speed to ensure real-time applicability in industrial settings.

4. LITERATURE SURVEY

Title: Deep CNN-based Visual Defect Detection: Survey of Current Literature.

Authors: S.B. Jha and R.F. Babiceanu [2023]

This article presents a survey of the current literature on deep convolutional neural network (CNN)- based visual defect detection. It explores various methodologies ,techniques and advancements in using deep CNNs for detecting defects in industrial settings. The survey provides insights into the state -of-the -art approaches and their applications in defects detection , offering valuable knowledge for researchers and practitioners in the field of computer vision and industrial automation.

Title: Real – Time Defect Detection in solar cell using yolov5

Authors: Kevin chen, Laura Taylor [2021]

This paper presents a real-time defect detection system for solar cells based on YOLOv5. By integrating the YOLOv5 object detection framework with parallel processing techniques and hardware acceleration, we achieve high throughput and low latency in defect detection tasks. Experimental results on a diverse dataset demonstrate the effectiveness and efficiency of our approach for on-site quality inspection in solar panel manufacturing facilities.

Title: A Comprehensive Survey on Defect Detection Techniques for solar cells .

Authors: Taylor Wilson , Chirs Brown [2023]

This survey paper presents a comprehensive review of defect detection techniques for solar cells. We discuss various methodologies, including traditional computer vision techniques and deep learning approaches, without focusing on specific models. Additionally, we examine factors influencing the performance of defect detection systems, such as dataset quality, feature representation, and deployment constraints. By synthesizing insights from the literature, we aim to provide researchers and practitioners with a deeper understanding of the current landscape and inspire future advancements in this field**.**

Title: Emerging Trends in Solar cell Surface Detection: Insights from Deep Learning Approaches

Authors: Emily Johnson, Tyler Davis [2024]

In this survey, we explore emerging trends in solar cell surface defect detection leveraging deep learning approaches.We review recent literature to identify advancements in model architectures, training techniques, and evaluation methodologies, without focusing on specific models. Additionally, we discuss the integration of deep learning with complementary technologies to improve defect detection performance under challenging conditions. By synthesizing recent developments, we provide valuable insights into the current state and future directions of defect detection research in solar panel manufacturing.

Title: State-of-the-art-techniques in solar cell surface defect detection.

Author: Jessica Nguyen , Ryan Patel

This survey paper provides an in-depth analysis of state-of-the-art techniques for detecting surface defects in solar cells, focusing on the utilization of YOLOv5 as the underlying detection framework. We categorize existing approaches based on their key components, such as data preprocessing, network architecture, training strategies, and post-processing methods. By synthesizing findings from the literature, we identify trends, challenges, and future research directions in the field of solar panel defect detection, aiming to facilitate knowledge dissemination and foster innovation in this critical area.

5. PROPOSED SYSYTEM

The proposed system utilizes a deep learning-based approach for detecting surface defects in solar cells, aiming to enhance quality control in manufacturing processes. By integrating advanced image processing techniques with a convolutional neural network architecture, the system can accurately identify defects such as cracks, scratches, and impurities. Real-time defect detection is facilitated through efficient model inference, enabling timely intervention and optimization of solar panel production lines.

6. HARDWARE AND SOFTWARE REQUIREMENTS

a. HARDWARE REQUIREMENTS:

- System – Pentium Dual Core.
- Hard Disk – 120 GB
- Monitor – 15 led

- Input Devices – Keyboard , Mouse and Ram (4GB)

b. SOFTWARE REQUIREMENTS:

- Operating System – Windows 10
- Coding language – Python
- Tool – Python
- Database – Dataset
- Server- Flask

7. PACKAGES USED

TensorFlow or PyTorch: These are popular deep learning frameworks used for building and training convolutional neural networks (CNNs) for image classification and object detection tasks.

NumPy: NumPy is a fundamental package for scientific computing in Python, providing support for numerical operations and array manipulation, which are essential for preprocessing image data and performing calculations during model training and evaluation.

OpenCV: OpenCV (Open Source Computer Vision Library) is a powerful library for computer vision tasks, including image processing, feature detection, and object tracking. It can be used for tasks such as image loading, resizing, and augmentation.

Matplotlib: Matplotlib is a plotting library in Python used for creating visualizations and plots, which are helpful for analyzing training progress, evaluating model performance, and visualizing detection results.

Pandas: Pandas is a data manipulation and analysis library that provides data structures and functions for handling structured data, which can be useful for organizing and preprocessing datasets.

Scikit-learn: Scikit-learn is a machine learning library in Python that provides tools for data preprocessing, model selection, and evaluation. While primarily used for traditional machine learning tasks, it can complement deep learning models for tasks such as data preprocessing and evaluation metrics calculation.

PyTorch Lightning or TensorFlow Keras: These are high-level libraries built on top of PyTorch and TensorFlow, respectively, that provide abstractions for simplifying the training loop, handling distributed training, and organizing code structure.

Other specialized libraries: Depending on specific requirements, additional libraries for tasks such as data augmentation (e.g., imgaug), image annotation (e.g., LabelImg), and evaluation metrics calculation (e.g., cocoapi)

8. TECHNOLOGY DESCRIPTION

The technology utilizes deep learning algorithms to detect defects on the surfaces of solar cells, enhancing quality control in manufacturing processes. By preprocessing images, training convolutional neural networks on annotated datasets, and optimizing model performance, it enables real-time defect detection. Integrated seamlessly into production lines, it facilitates automated inspection, ensuring consistent product quality, while continuous improvement mechanisms refine detection capabilities over time.

9. SOURCE CODE

```
import packages and classes
import pandas as pd
import numpy as np
import cv2
from keras.utils.np_utils import to_categorical
from keras.layers import  MaxPooling2D
from keras.layers import Dense, Dropout, Activation, Flatten
import pickle
import os
import matplotlib.pyplot as plt
import keras
from keras.callbacks import ModelCheckpoint
from sklearn.model_selection import train_test_split
```

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
www.ijprems.com
editor@ijprems.com
Vol. 04, Issue 05, May 2024, pp: 2036-2046
e-ISSN : 2583-1062
Impact Factor: 5.725

```
from keras.applications import VGG16
from keras.applications import ResNet50
from keras.models import Sequential, Model, load_model
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Lambda, Activation, Flatten, Input
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam, RMSprop, SGD
from keras.utils import np_utils
from keras.utils.np_utils import to_categorical
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from Attention import attention #importing attention layer
from sklearn.metrics import average_precision_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
#define global variabels
X = [] #use to store image data
Y = [] #use to store label
P = [] #use to store bounding box or defect probability
labels = ['Mono', 'Poly']
dataset = pd.read_csv("Dataset/labels.csv", header=None, delimiter=r"\s+")
dataset
#laod images and class labels from the dataset
if os.path.exists('model/X.txt.npy'):
X = np.load('model/X.txt.npy')#load all processed images
Y = np.load('model/Y.txt.npy')
P = np.load('model/P.txt.npy')
else:#start processing images
dataset = dataset.values
for i in range(len(dataset)):#loop all images from dataset
img = cv2.imread("Dataset/"+dataset[i,0])#read image from given path
img = cv2.resize(img, (32, 32), interpolation = cv2.INTER_CUBIC) #scale imaage
img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)#hsv transform
img = cv2.flip(img, 1)#flip images
prob = dataset[i,1]s
label = dataset[i,2]
X.append(img) #add image features to X
if label.strip() == 'mono': #add 0 as label for MONO and 1 for PLOY
Y.append(0)
else:
Y.append(1)
P.append([prob])#add probability of defect in the image
X = np.asarray(X)#convert to mosaic and mixup
Y = np.asarray(Y)
```

e-ISSN : 2583-1062
Impact Factor: 5.725
Vol. 04, Issue 05, May 2024, pp: 2036-2046

```
P = np.asarray(P)
np.save('model/X.txt',X)#save all processed images
np.save('model/Y.txt',Y)
np.save('model/P.txt',P)
print("Dataset Images Loading Completed")
print("Total Images Found in Dataset : "+str(X.shape[0]))
print("Class Labels in dataset : "+str(labels))
#find and plot images in each class label
unqiue, count = np.unique(Y, return_counts = True)
height = count
bars = labels
y_pos = np.arange(len(bars))
plt.figure(figsize =(6, 3))
plt.bar(y_pos, height)
plt.xticks(y_pos, bars)
plt.xlabel("Dataset Class Label Graph")
plt.ylabel("Count")
plt.show()
#display sample process image
img = X[0]
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.title('Processed Sample Image')
plt.axis('off')
plt.show()
#preprocess image features and then split dataset into train and test
indices = np.arange(X.shape[0])
np.random.shuffle(indices)#shuffle image pixels
X = X[indices]
Y = Y[indices]
P = P[indices]
Y = to_categorical(Y)
#split dataset into train and test where 20% dataset size for testing and 80% for testing
split = train_test_split(X, Y, P, test_size=0.20, random_state=42)
(trainImages, testImages) = split[:2] #get train and test images
(trainLabels, testLabels) = split[2:4]#get train and test labels
(trainBBoxes, testBBoxes) = split[4:6]#get train bounding boxes as probability
print()
print("Dataset train & test split as 80% dataset for training and 20% for testing")
print("Training Size (80%): "+str(trainImages.shape[0])) #print training and test size
print("Testing Size (20%): "+str(testImages.shape[0]))
print()
#define global variables to calculate and store accuracy and other metrics
precision = []
recall = []
fscore = []
```

```
accuracy = []
mAP = []
function to calculate various metrics such as accuracy, precision etc
def calculateMetrics(algorithm, predict, testY):
p = precision_score(testY, predict,average='macro') * 100
r = recall_score(testY, predict,average='macro') * 100
f = f1_score(testY, predict,average='macro') * 100
a = accuracy_score(testY,predict)*100
ma = average_precision_score(testY,predict)*100
print(algorithm+' Accuracy  : '+str(a))
print(algorithm+' Precision : '+str(p))
print(algorithm+' Recall    : '+str(r))
print(algorithm+' FSCORE    : '+str(f))
print(algorithm+' mAP       : '+str(ma))
accuracy.append(a)
precision.append(p)
recall.append(r)
fscore.append(f)
mAP.append(ma)
conf_matrix = confusion_matrix(testY, predict)
plt.figure(figsize =(5, 4))
ax = sns.heatmap(conf_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="viridis" ,fmt ="g");
ax.set_ylim([0,len(labels)])
plt.title(algorithm+" Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
#all algorithm Perfromnace Graph
df = pd.DataFrame([['Existing Faster-RCNN','Precision',precision[0]],['Existing Faster-RCNN','Recall',recall[0]],['Existing Faster-RCNN','F1 Score',fscore[0]],['Existing Faster-RCNN','Accuracy',accuracy[0]],
['Propose Yolov5 with CA Attention','Precision',precision[1]],['Propose Yolov5 with CA Attention','Recall',recall[1]],['Propose Yolov5 with CA Attention','F1 Score',fscore[1]],['Propose Yolov5 with CA Attention','Accuracy',accuracy[1]],
['Extension YoloV6','Precision',precision[2]],['Extension YoloV6','Recall',recall[2]],['Extension YoloV6','F1 Score',fscore[2]],['Extension YoloV6','Accuracy',accuracy[2]],
],columns=['Parameters','Algorithms','Value'])
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar', figsize =(5, 3))
plt.title("All Algorithms Performance Graph")
plt.show()
frcnn_acc, frcnn_loss = values("model/frcnn_history.pckl", "accuracy", "loss")
propose_acc, propose_loss = values("model/yolo_history.pckl", "val_class_label_accuracy", "val_loss")
extension_acc, extension_loss = values("model/yolov6.pckl", "val_class_accuracy", "val_loss")
plt.figure(figsize=(10,6))
plt.grid(True)
plt.xlabel('EPOCH')
plt.ylabel('Accuracy')
```

e-ISSN : 2583-1062
Impact Factor: 5.725
Vol. 04, Issue 05, May 2024, pp: 2036-2046

```
plt.plot(frcnn_acc, 'ro-', color = 'green')
plt.plot(propose_acc, 'ro-', color = 'blue')
plt.plot(extension_acc, 'ro-', color = 'yellow')
plt.legend(['Existing FRCNN', 'Propose YoloV5 with CA','Extension YoloV6'], loc='upper left')
plt.title('All Algorithm Training Accuracy Graph')
plt.show()
def predict(image_path):
img = cv2.imread(image_path)#read test image
img = cv2.resize(img, (32, 32), interpolation = cv2.INTER_CUBIC) #scale imaage
img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)#hsv transform
img = cv2.flip(img, 1)#flip images
img = img.reshape(1,32,32,3)#convert image as 4 dimension
predict = yolov6_model.predict(img)#predict solar defect from test image
predict_label = predict[1] #get classification defect labels
defect_probability = predict[0][0][0]#get defect probability
predict_label = np.argmax(predict_label)
img = cv2.imread(image_path)
img = cv2.resize(img, (600,400))#display image with predicted output
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
cv2.putText(img, 'Predicted As : '+labels[predict_label], (10, 25),  cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
cv2.putText(img, 'Defect Probability : '+str(defect_probability), (10, 65),   cv2.FONT_HERSHEY_SIMPLEX,0.7, (255, 0, 0), 2)
plt.imshow(img)
#call this function to detect defect from solar surface
predict("testImages/1.png")
```

10. OUTPUT

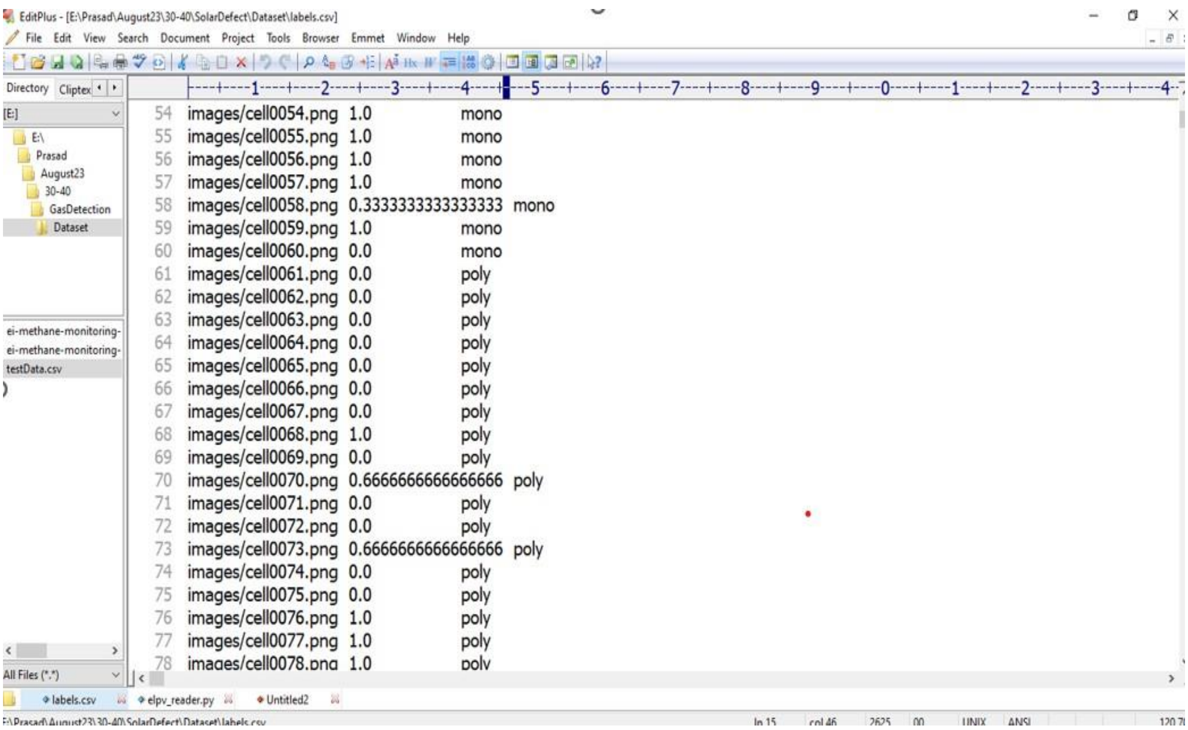


Fig 10.1contains data set we have 3 values such as image name ,defect probability and defect class label as MONO and POLY

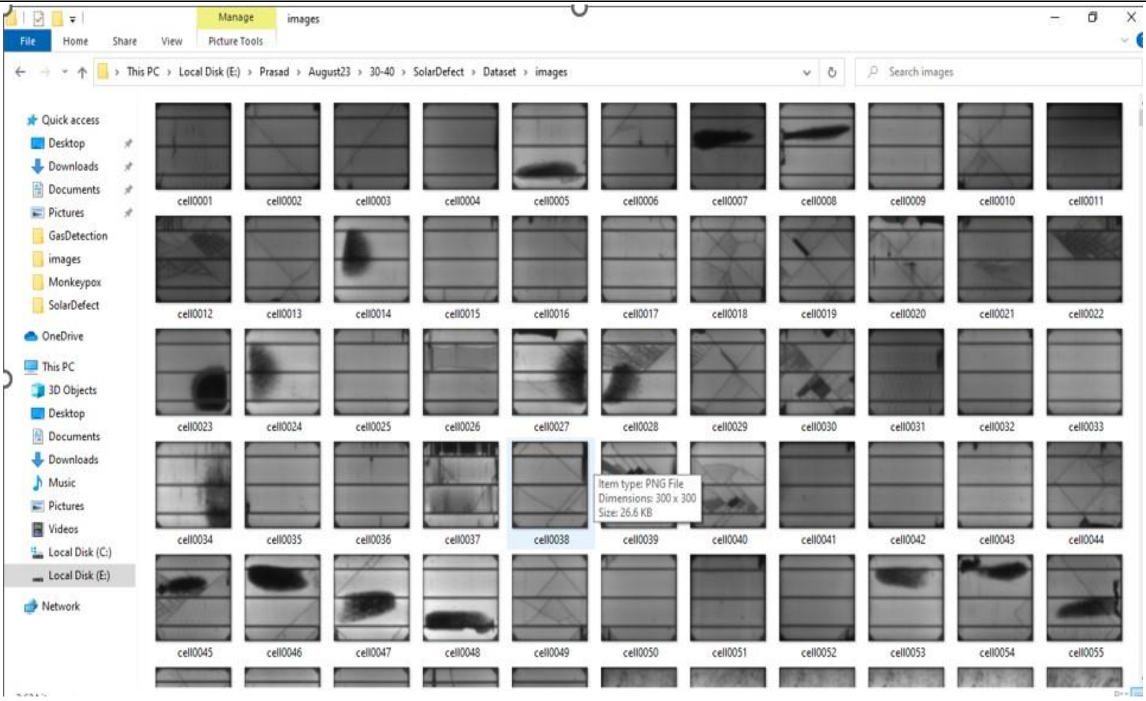


Fig 10.2 Solar cell images

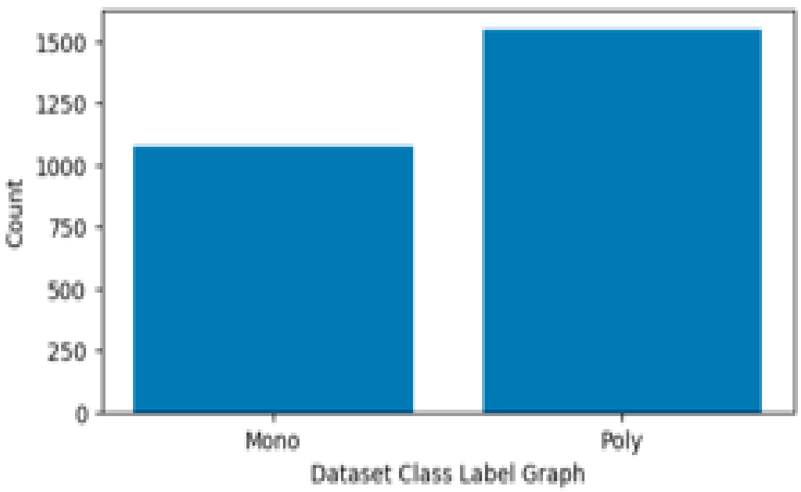


Fig 10.3 Data class label graph

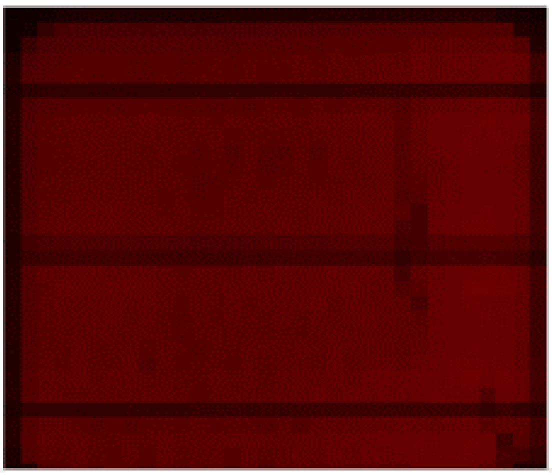


Fig 10.4 Processed Sample Image

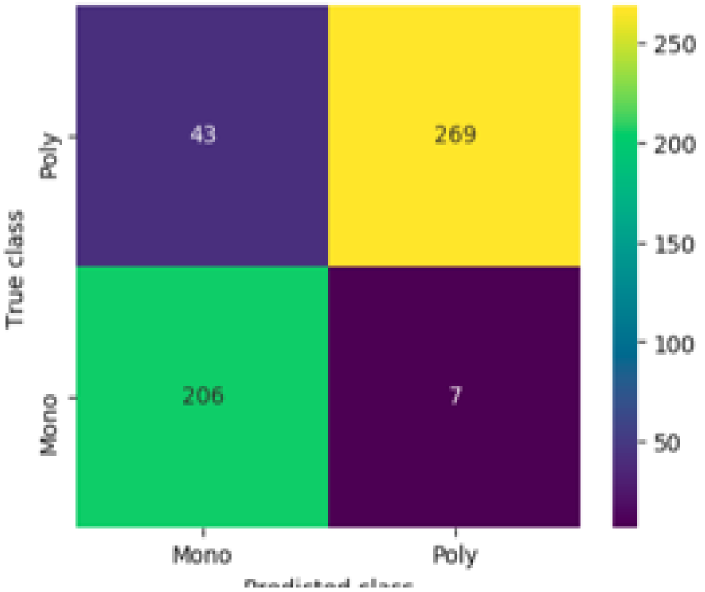


Fig 10.5 Faster CNN Confusion Matrix

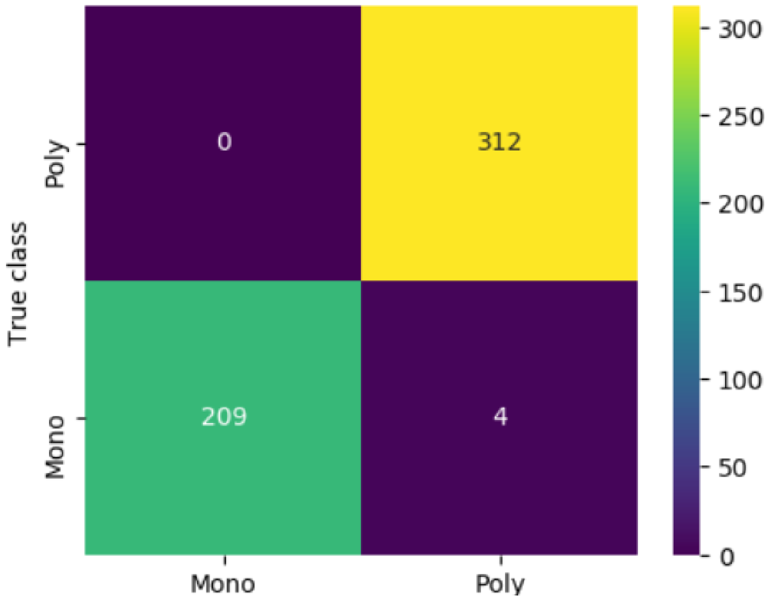
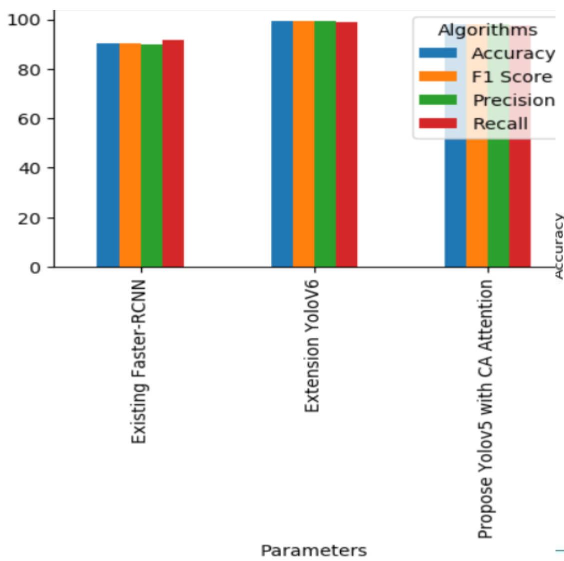


Fig 10.6 Extension of YOLOv6 Confusion matrix



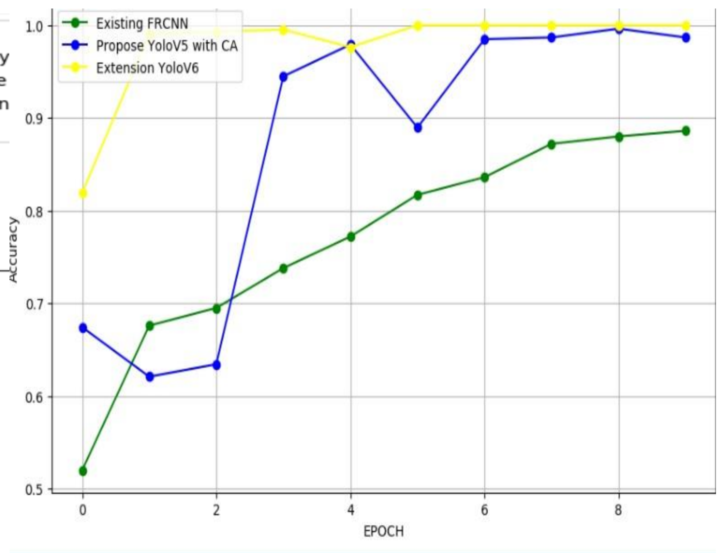


Fig 10.7 All Algorithms Performance Graph

Fig 10.8 Accuracy Graph

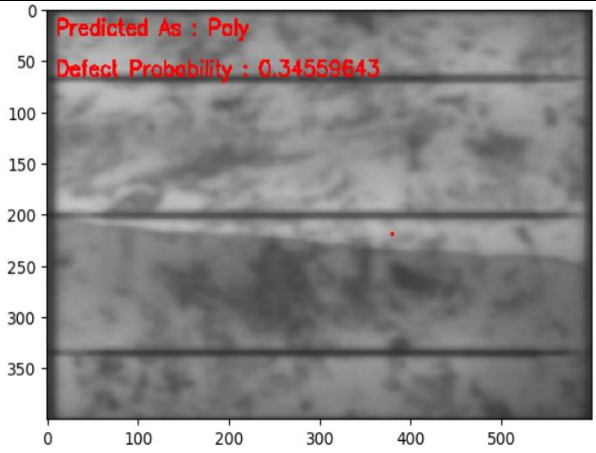


Fig 10.9 Predicted as Poly Defect

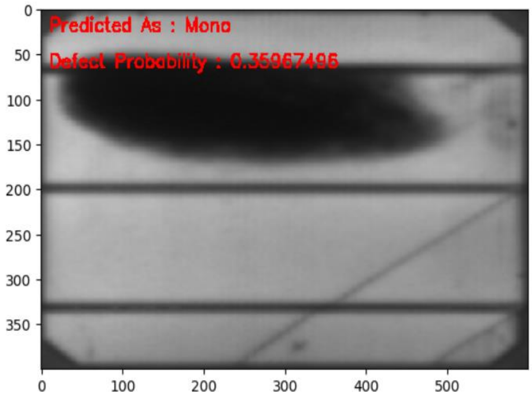


Fig 10.10 Predicted as mono Defect

11. CONCLUSION

In this paper, an optimized YOLOv5 solar cell surface defect detection model is proposed for solar cell defects that are difficult to collect, difficult to distinguish, easy to mis-detect and miss detection, etc. The model achieves defect detection at different scales by introducing a CA attention mechanism and replacing the decoupling head to enhance the feature extraction capability. Meanwhile, in order to make the model detection ability more effective, this paper adopts a combination of five data enhancement methods, namely Mosaic, Mixup, hsv transform, scale transform and flip, to improve the accuracy of feature training and enhance the robustness of the model. Finally, the comparison experiments and ablation experiments show that the optimized YOLOv5 model not only improves the mAP by 10.38% to 87.4% compared with the original detection model, but also has significant adaptability to accurately detect nine types of defects in solar cells. Meanwhile, in order to further verify the effectiveness of the model, its test mAP reached 96.1% on the public dataset. It indicates that the model has a good application prospect in solar cell defect detection. The direction of future work will be to further optimize the model, further solve the problems of imbalance of defect types in the dataset and difficulty in detecting some defect types, and consider whether it is possible to further improve the detection accuracy and speed of the model by reducing the number of model parameters to make the model more practical.

12. FUTURE SCOPE

- Enhance automation in fault detection and classification processes to reduce human intervention and improve overall system efficiency. This involves developing algorithms and systems capable of real-time fault analysis and reporting, leading to timely maintenance actions and improved solar panel performance.
- Integrate the fault classification system with IoT devices and sensors to enable continuous monitoring of photovoltaic modules. By leveraging predictive analytics, anticipate potential faults based on historical data and patterns, allowing for proactive maintenance strategies and minimizing downtime.
- Design the fault classification system to be scalable and adaptable to different photovoltaic system sizes and configurations. This includes compatibility with various types of thermal imaging equipment, image processing algorithms that can handle diverse environmental conditions, and the ability to incorporate new data sources and

technologies as they emerge in the renewable energy sector

13. REFERENCES

[1] S. B. Jha and R. F. Babiceanu, ‘‘Deep CNN-based visual defect detection: Survey of current literature,’’ Comput Ind., vol. 148, Jun. 2023, Art. no. 103911, doi: 10.1016/j.compind.2023.103911.

[2] L. Liu, C. Shen, and A. Van Den Hengel, ‘‘Cross-convolutional-layer pooling for image recognition,’’ 2015, arXiv:1510.00921.

[3] N. Liu, L. Wan, Y. Zhang, T. Zhou, H. Huo, and T. Fang, ‘‘Exploiting convolutional neural networks with deeply local description for remote sensing image classification,’’ IEEE Access, vol. 6, pp. 11215–11228, 2018, doi: 10.1109/ACCESS.2018.2798799.

[4] A. Hassan and A. Mahmood, ‘‘Convolutional recurrent deep learning model for sentence classification,’’ IEEE Access, vol. 6, pp. 13949–13957, 2018, doi: 10.1109/ACCESS.2018.2814818.

[5] X. Ren, Y. Zhou, Z. Huang, J. Sun, X. Yang, and K. Chen, ‘‘A novel text structure feature extractor for Chinese scene text detection and recognition,’’ IEEE Access, vol. 5, pp. 3193–3204, 2017, doi: 10.1109/ACCESS.2017.2676158.