SOLAR CELL DEFECT DETECTION BASED ON OPTIMIZED YOLOV5

#  **Dr Ch.Vijayakumar1, Changalwala Kameswara Vishnu Vallabha2, Chebrolu Bangaru Sai Prasanth 3, Potlacheruvu Sanjay 4,Palla Rahul5**

Associate. Professor, CSE Dept, ACE Engineering College, Hyderabad, India

vijay.chandarapu@gmail.com

2Student, CSE Dept, ACE Engineering College, Hyderabad, India

vishnuvallabh2428@gmail.com

3 Student, CSE Dept, ACE Engineering College, Hyderabad, India

Chbsp2002@gmail.com

4 Student, CSE Dept, ACE Engineering College, Hyderabad, India

sanjaypotlacheruvu@gmail.com

5 Student, CSE Dept, ACE Engineering College, Hyderabad, India

Guptharahul45@gmail.com

***Abstract :***

The project begins by collecting and curating extensive datasets comprising. 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**

Essentially, this project will be able to Detect Detection based on optimized yolov5 to identify the accuracy of the solar cell. 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
* Optimizations
* 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.

**4. LITERATURE SURVEY**

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

AUTHOR: S.B. Jha and R.F. Babiceanu.

YEAR: 2023

DESCRIPTION: 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

AUTHOR: Kevin chen, Laura Taylor

YEAR: 2021

DESCRIPTION:

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.

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

AUTHOR: Emily Johnson, Tyler Davis

YEAR: 2024

DESCRIPTION:

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.

**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**

 **6.1 HARDWARE REQUIREMENTS:**

* Processor: Min. Core i3 processor ,Pentium Dual core
* RAM: 2GB (Min.) or 8GB (Recommended)
* Hard Disk Space: 50GB+

 **6.2 SOFTWARE REQUIREMENTS:**

* Programming Language: Python
* Operating System: Windows 7 or later versions

of windows.

**7. PACKAGES USED**

**TensorFlow or PyTorch Lightning**

TensorFlow is a popular open-source Python machine learning toolkit for creating and training deep neural networks. It has a versatile architecture and supports a variety of platforms, including CPU, GPU, and TPU. TensorFlow simplifies the implementation of complicated algorithms and models, allowing developers to create scalable and efficient machine learning systems.

**Keras**

Keras is a Python-based high-level neural network API that operates on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. It offers an easy-to-use interface for building and training deep learning models, letting users to easily experiment with alternative architectures and hyperparameters. Keras also provides pre-trained models as well as a huge collection of building blocks for developing sophisticated models.

 **Open CV**

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.

**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.

**Numpy**

NumPy is an important Python package for scientific computation. It supports huge, multidimensional arrays and matrices, as well as a diverse collection of high-level mathematical operations for these arrays. NumPy is a popular choice for numerical operations in scientific research and data analysis due to its efficient and user-friendly interface.

**Pandas**

Pandas is a popular open-source Python data analysis and manipulation package. It offers sophisticated data structures and tools for working with structured data, including as data frames and series, and it allows for quick data processing, cleaning, merging, and reshaping. Pandas also supports reading and writing a variety of file types, including CSV, Excel, and SQL databases.

**Matplotlib**

Matplotlib is a popular Python data visualization package. It includes line graphs, scatter plots, bar plots, and histograms among its 2D and 3D displays. Matplotlib is a useful tool for data exploration and communication since it is extremely customizable and supports extensive labelling, annotations, and text formatting.

 **OS and time**

The 'os' module in Python allows you to interact with the operating system. It has functions for creating and removing folders, manipulating files, and changing environment variables. The 'time' module in Python contains methods for working with time-related actions. It has functions for obtaining the current time, postponing program execution, and converting between several time formats.

**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

**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

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)

 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()

# prediction visualization

plt.imshow(np.log(confusion\_matrix(y\_test,y\_pred)),cmap = 'Blues',interpolation = 'nearest')

plt.ylabel('True')

plt.xlabel('Predicted')

 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')

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 10.1: Dataset images



 Fig 10.2: Solar cell images



Fig 10.3: Data Class Label Graph



Fig 10.4:Processed Sample Image



Fig 10.5: matrix yolov5



Fig 10.6:Extensive of Yolov6 confusion matrix

Fig 10.8: Accuracy Graph



 Fig 10.9 Predicted as Poly Defect



Fig 11: 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. And by this to get a popular accuracy of the defect data we have used the extensive version of yolov5 that v6. It is called as but it provide 99.9% of accuracy .

**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.