International Journal of Research in Vocational Studies (IJRVOCAS)
Vol. 5, No. 2, August 2025, pp. 33~49
Print ISSN 2777-0168| Online ISSN 2777-0141| DOI prefix: 10.53893
https://journal.gpp.or.id/index.php/ijrvocas/index

Implementation of Convolutional Neural Network in Mobile Applications for Solar Panel Crack and Efficiency Prediction

Wisnu Kurniawan Sodiq[1, *], Ahmad Taqwa[2] & Kusumanto[3]

[1] Magister Program, Renewable Energy Engineering, Politeknik Negeri Sriwijaya, Palembang, Indonesia
[2,3] Renewable Energy Engineering, Politeknik Negeri Sriwijaya, Palembang, Indonesia

Email address:

wisnu.ks@gmail.com
[*]Corresponding author

To cite this article:

Sodiq, W. K., Taqwa, A., & Kusumanto. Implementation of Convolutional Neural Network in Mobile Applications for Solar Panel Crack and Efficiency Prediction. International Journal of Research in Vocational Studies (IJRVOCAS), 5(2), 33–49. https://doi.org/10.53893/ijrvocas.v5i2.430

Received: 05 25, 2025; **Revised**: 06 15, 2025; **Accepted**: 07 05, 2025; **Published**: 08 25, 2025

IJRVOCAS is licensed under a Creative Commons Attribution 4.0 International License.

Abstract: Solar panels, as a renewable energy source, are susceptible to efficiency degradation due to cracks in solar cells. Manual crack detection has many limitations, while the use of specialized tools like electroluminescence imaging is not economical for small-scale users. Therefore, this research aims to develop an image-based automatic detection system using the Convolutional Neural Network (CNN) method, specifically the YOLOv8 architecture, integrated into a web-based mobile application using the Flask framework. Solar panel image datasets were collected and annotated using Roboflow, then trained in Google Colab with the help of a GPU. The trained model is integrated into a web-based mobile application, allowing users to upload panel images, detect cracked areas, and estimate panel efficiency based on linear regression of the crack area. Testing results show that the system can function in real-time, although the accuracy of efficiency estimation can still be improved due to limitations in data quantity and variation. This research is expected to be an economical and practical solution for solar panel monitoring.

Keywords: Convolutional Neural Network, Solar Panel, Crack, Efficiency, Mobile Application

1. Introduction

1.1. Background

Solar energy is a renewable energy source with great potential to reduce dependence on fossil fuels [1]. Solar panels, as the primary device in photovoltaic systems, convert sunlight into electricity [2]. The reliability and efficiency of solar panels are largely determined by the physical condition of the modules, especially the integrity of their constituent solar cells [3]. One of the main problems often occurring in solar panels is cracks on the surface of the solar panel [4]. These cracks, whether micro or macro, can disrupt the flow of electricity and cause a significant decrease in power output [5]. This efficiency reduction not only impacts the overall system performance but also has the potential to shorten the panel's lifespan [6].

Crack detection is crucial as part of solar panel system maintenance and optimization efforts [7]. Conventional detection methods like Electroluminescence (EL) imaging and thermography can accurately detect cracks, but these methods have limitations in terms of cost and accessibility [8]. Therefore, an alternative method that is more practical and

economical is needed, especially for applications in home user environments or small-scale installations [9]. Recent advancements in deep learning, particularly the use of algorithms like DANet, offer promising solutions for efficient crack detection in solar panels, enhancing maintenance and optimizing performance [10].

With the development of artificial intelligence technology, particularly in digital image processing, Convolutional Neural Network (CNN) has shown excellent performance in recognizing visual patterns and detecting objects in images [11]. One popular and efficient CNN architecture is YOLO (You Only Look Once), which has now evolved to YOLOv8 [12]. This algorithm is known for being fast and accurate in object detection, including small objects like cracks [13]. By utilizing YOLOv8 as a detection model and integrating it into a Flask-based web application, users can upload solar panel images via mobile devices or computers and directly obtain visual information about the panel's condition [14]. Additionally, this system is designed to estimate panel efficiency based on the detected crack area through a linear regression model [15]. This approach is expected to make solar panel inspection self-contained, fast, and inexpensive, thereby helping to raise awareness of photovoltaic system maintenance and supporting renewable energy sustainability at various scales of use [14]. This innovative solution not only enhances the crack detection process but also empowers users to take proactive measures in maintaining their solar panels for optimal performance [10].

1.2. Problem Formulation

Based on the background explained previously, this research focuses on answering several fundamental questions related to the automatic detection of cracks in solar panels and the prediction of their efficiency. The problem formulations in this research are as follows:

- How to design and train a CNN model (YOLOv8) capable of detecting cracks in solar panels?
- How to integrate this detection model into a Flask-based web application that is easily accessible via mobile devices?
- How to develop a method for predicting solar panel energy efficiency degradation based on the automatically detected crack area?

1.3. Research Objectives

Based on the problem formulation, this research aims to develop an automatic crack detection system for solar panels and predict their energy efficiency. This objective is realized through several technical and strategic steps focused on utilizing deep learning technology and web-based application development. In detail, the objectives of this research are as follows:

- To develop a Convolutional Neural Network (CNN) model using the YOLOv8 architecture to detect cracks in solar panel images.
- To build a Flask-based web application accessible via mobile and desktop devices to display real-time detection results.
- To implement a linear regression method into the system to calculate the estimated efficiency degradation of solar panels based on the detected crack area.

1.4. Research Benefits

This research is expected to make a positive contribution in the field of renewable energy, particularly in the aspects of maintenance and monitoring of solar panel conditions. The benefits of this research can be elaborated as follows:

- For Institutions: As a form of academic contribution to the development of artificial intelligence technology for renewable energy, especially in image-based monitoring and detection of solar panel conditions.
- For Science and Technology Development: To serve as a scientific and technical reference in the implementation of deep learning (CNN) and its integration into web applications for image processing in the energy sector.
- For Society: To provide a practical, economical, and easily accessible solution for inspecting solar panel conditions without requiring specialized equipment, thereby raising awareness of the importance of solar panel maintenance for optimal energy efficiency.

1.5. Hypothesis

Based on the problem formulation and theoretical framework presented, the hypotheses in this study can be formulated as follows:

- The Convolutional Neural Network (CNN) model based on YOLOv8 can be used to detect cracks in solar panels through digital image analysis.
- Solar panel efficiency can be automatically estimated using a linear regression approach based on the detected crack area.
- The Flask-based web application can integrate detection and efficiency estimation processes in real-time and be well-accessed via both desktop and mobile devices.

1.6. Novelty

This research offers several novelties that add academic and implementational value:

- Integration of CNN YOLOv8 in Web Applications: This research integrates a Convolutional Neural Network (CNN) model based on YOLOv8 into a crack detection system for solar panels in the form of a web application. This differs from previous research, such as that conducted by Hassan and Dhimish (2023), which mostly discussed CNN-based image classification approaches without direct implementation into user applications. Furthermore, conventional approaches, like those used by Dhimish and Holmes (2019), still rely on electroluminescence-based visual detection methods that require specialized and less accessible tools.

- Efficiency Prediction Feature: Second, this system not only provides visual crack detection but also implements a solar panel efficiency prediction feature based on the detected crack area. This process is carried out using a simple linear regression approach directly connected to the YOLOv8 detection results. This approach provides predictive value that is beneficial for users to make technical decisions regarding panel conditions.
- Multi-Device Web-Based Application: Third, the developed application is web-based and designed to be usable on various devices, including mobile devices, with an interface that remains functional and easy to use on small screens, without requiring specific mobile application installations. This provides high flexibility in use, both in technical and non-technical environments.
- No Special Supporting Devices Required: Fourth, the system does not require special supporting devices such as EL (Electroluminescence) cameras or thermal cameras; it simply uses images from a standard smartphone camera. Thus, this system becomes a more practical and economical solution, especially for monitoring solar panels on a small to medium scale.

1.7. Research Framework

The research framework is designed to provide a comprehensive overview of the logical flow of problem-solving, from identification to technology-based solution development. This research is motivated by real-world problems, namely the difficulty of manually detecting cracks in solar panels. Visual detection by field operators is often inaccurate, and special tool-based methods like electroluminescence imaging or thermography are costly and require inaccessible equipment.

As an alternative solution, this research utilizes Convolutional Neural Network (CNN) technology, which has proven effective in image processing and object detection. The YOLOv8 model was chosen as the CNN architecture due to its ability to detect small objects in real-time, as well as its speed and computational efficiency. To build this model, a solar panel image dataset was collected from field documentation and open sources, then annotated using Roboflow to differentiate between cracked and non-cracked panels. The model training process was carried out in Google Colab, leveraging GPUs for faster and more efficient training. The model was trained to recognize crack areas in solar panel images, and the model results were saved in .pt format for use in the application.

After training was completed, the model was integrated into a web-based application built using the Flask framework. This application allows users to upload solar panel images through a browser (both desktop and mobile), process these images, and display visual crack detection results. In addition to crack detection, the system also features an estimated solar panel efficiency calculation based on the percentage of the detected crack area. This calculation is performed using a simple linear regression model. Efficiency estimation is calculated based on the linear relationship between the crack area and solar panel performance, analyzed from initial test data.

2. Literature Review

2.1. Introduction to Solar Panels

The flue gas for the experiment was generated from the carbonization of coconut shells [16]. The absorbent solution was prepared using technical grade potassium carbonate (K_2CO_3), Diethanolamine (DEA), and aquadest [17].

Solar panels are among the most popular and efficient renewable energy technologies for converting solar energy into electricity [18]. The use of solar panels as an alternative energy source is increasing in line with the global need for environmentally friendly and sustainable energy sources [19]. This technology utilizes the photovoltaic effect, where solar cells made of semiconductor materials, such as silicon, capture photons from sunlight and convert them into an electric current [20]. The main advantage of solar panels is their ability to generate electricity without greenhouse gas emissions and other pollutants, thereby contributing to the reduction of negative environmental impacts [21].

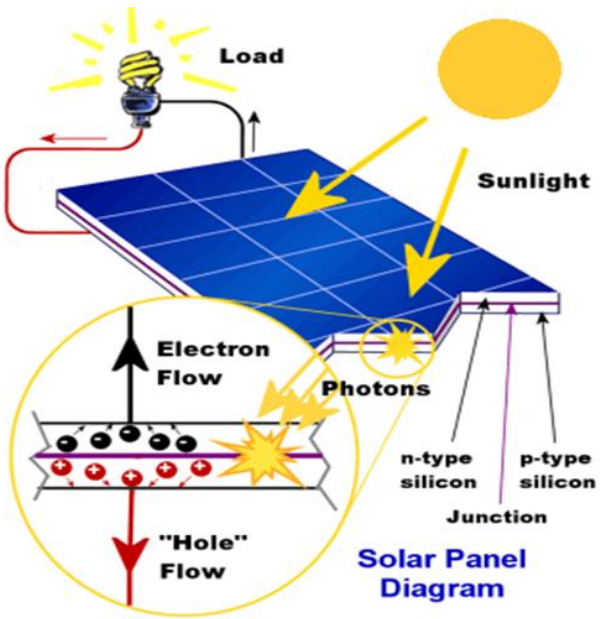

Figure 1. *How Solar Panels Work*

The configuration of series and parallel connections in solar panels plays a crucial role in determining the overall performance of the photovoltaic system [22]. In series connections, several cells or solar modules are connected in sequence to increase the total system voltage, while in parallel connections, several modules are connected side-by-side to increase the total current [23]. Proper configuration selection can help optimize the power output of the photovoltaic system according to specific application needs [24]. This optimization is essential for enhancing energy efficiency and ensuring that the system meets the load requirements effectively [25].The combination of series and parallel connections allows for flexibility in adapting to varying sunlight conditions, ultimately maximizing energy production from the solar

panels [22].Efficient configuration of solar panels not only enhances voltage and current output but also mitigates the impact of environmental factors such as shading and temperature variations [26].

Figure 2. *Cracked Solar Panel and Good Solar Panel*

However, the problem of cracks in solar panels becomes a significant challenge that can affect the efficiency and longevity of the system [27]. Cracks in solar panels can be caused by various factors, including mechanical stress, extreme temperature changes, and low material quality [28]. The impact of these cracks can be very detrimental, ranging from a reduction in the active area for sunlight absorption to the occurrence of hotspots that can further damage the module [29]. Therefore, crack detection and mitigation are important aspects in the maintenance and optimization of solar panel efficiency [30]. Effective strategies, such as the use of deep learning models for crack detection, can significantly enhance maintenance efforts and prolong the lifespan of solar panels [31].

2.2. Problem of Cracks in Solar Panels

Solar panels are key components in photovoltaic systems that convert solar energy into electricity. Although this technology offers a sustainable solution for energy needs, its reliability can be affected by cracks in the solar cells. These cracks can occur during the manufacturing process, installation, or daily operation, and have a significant impact on the efficiency and lifespan of solar panels. The causes of cracks in solar panels include:

2.2.1. Mechanical Stress

Mechanical stress is one of the primary causes of cracks in solar panels and can occur during various stages, including production, transportation, installation, and operation. Some sources of mechanical stress include:

- Heavy Loads: Solar panels that are subjected to heavy loads during transportation or installation can cause cracks. For example, if panels are stacked in high piles without adequate support, the weight of the panels above can press down on those below, causing damage.
- Strong Winds and Heavy Rain: The pressure generated by strong winds or heavy rain impacting the panel surface can cause mechanical stress and cracks. Panels installed in areas with extreme weather conditions must be designed to withstand wind and heavy rain loads to prevent damage.
- Vibration and Shocks: Vibrations and shocks occurring during transportation can cause micro-cracks in solar panels. Inadequate packaging or rough handling methods during shipping can exacerbate this problem.
- Improper Handling: During installation, solar panels that are not handled correctly, such as being dropped or bent, can experience mechanical stress leading to cracks.

2.2.2. Extreme Temperature Changes

Extreme temperature changes are another factor that can cause cracks in solar panels. Some important aspects of thermal stress include:

- Material Expansion and Contraction: Materials in solar panels, especially silicon semiconductors, expand when heated and contract when cooled. Extreme temperature changes can cause uneven material expansion and contraction, leading to thermal stress and cracks.
- Day-Night Cycles: Repeated day-night cycles cause solar panels to experience continuous temperature fluctuations. During the day, panels are exposed to direct sunlight and heat up, while at night, they cool down. The thermal stress resulting from these cycles can trigger micro-cracks that worsen over time.
- Exposure to Extreme Temperatures: Solar panels installed in environments with very high or very low temperatures, such as deserts or high mountains, are more susceptible to cracks due to repeated and intense material expansion and contraction.

2.2.3. Material and Manufacturing Quality

Material quality and manufacturing processes play a significant role in the resistance of solar panels to cracking. Some factors related to quality include:

- Low-Quality Materials: The use of low-quality semiconductor materials or other materials that do not meet standards can increase the risk of cracking. Materials with low mechanical strength are more prone to damage from mechanical or thermal stress.
- Manufacturing Defects: Defects occurring during the manufacturing process, such as poor soldering, uneven semiconductor layers, or the presence of inclusions in the material, can be starting points for cracks. Poor soldering quality, for example, can lead to weak connections between solar cells, making them more susceptible to cracking.
- Inconsistent Manufacturing Processes: Inconsistencies in manufacturing processes, such as variations in processing temperature or pressure, can lead to material non-uniformity, increasing the risk of cracking. Strict quality control standards must be applied to ensure that each panel is produced to the same specifications and meets high-quality standards.

2.3. Convolutional Neural Network (CNN) for Image Processing

Convolutional Neural Network (CNN) is a type of deep learning architecture that is highly effective for tasks involving grid-structured data, such as images. CNN was first popularized by Yann LeCun in the late 1980s with the development of LeNet, an early CNN model used for handwritten character recognition. Since then, CNN has rapidly evolved and become the foundation for many visionary applications such as face recognition, image classification, image segmentation, and object detection.

2.3.1. CNN Architecture

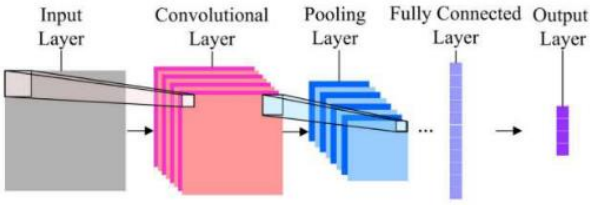

Figure 3. *CNN Layer Model*

The CNN architecture consists of several types of layers that work synergistically to extract features from input images and perform classification. The main components of CNN include:

- Convolutional Layer: The core of CNN is responsible for extracting features from the input image. In this layer, filters (also known as kernels) are applied to the input image to produce feature maps. These filters move across the image with a specific stride and apply convolution operations to detect various features such as edges, corners, and textures.
- Pooling Layer: Used to reduce the dimensions of feature maps, which helps in reducing the number of parameters and computational complexity. Pooling also helps prevent overfitting by reducing the network's sensitivity to shifts and distortions. The most common types of polling are max pooling, which takes the maximum value in each pooling area, and average pooling, which takes the average value in each pooling area.
- Activation Layer: An activation function is applied after the convolution operation to introduce non-linearity into the model. The most used activation function is the Rectified Linear Unit (ReLU).
- Normalization Layer: Batch normalization is a technique used to normalize the output of the previous layer, ensuring that each batch has a stable distribution. This helps in speeding up the training process and improving network stability.
- Fully Connected Layer: At the end of the CNN, the fully connected layer connects every neuron from the previous layer to every neuron in this layer. This layer performs classification based on the features that have been extracted and processed by previous layers. The output of this layer is usually passed to a softmax function to generate class probabilities.

2.3.2. CNN Training Process

Training a CNN involves using the backpropagation algorithm to optimize the filter weights and other parameters within the network. This algorithm calculates the gradients of the loss function (such as cross-entropy for classification) with respect to the network parameters and updates these parameters using optimization algorithms like stochastic gradient descent (SGD) or Adam optimizer.

- Forward Propagation: Input data (images) are passed through the network from the input layer to the output layer. In each layer, features are extracted and processed, resulting in the final output, which is the network's prediction.
- Loss Function: A loss function is used to measure how well the network's prediction compares to the true labels. One loss function often used for classification is categorical cross-entropy.
- Backward Propagation: After calculating the loss, backward propagation is used to update the network's weights. The backpropagation algorithm calculates the gradients of the loss function with respect to each weight in the network and uses optimizers such as stochastic gradient descent (SGD) or Adam optimizer to update the weights.
- Optimizers: Optimizers are algorithms used to update network weights by minimizing the loss. SGD and Adam are two of the most used optimizers. Adam is an adaptive optimizer that adjusts the learning rate for each parameter, which often speeds up convergence.

2.3.3. Applications of CNN in Image Processing

CNNs have been applied in various image processing applications with very satisfactory results. Some key applications include:

2.3.3.1. Object Recognition:

- CNNs are used to classify images into predefined categories. Models like AlexNet, VGGNet, and ResNet have achieved outstanding results in the ImageNet competition, a well-known benchmark in object recognition.
- AlexNet: Object recognition with a deep architecture and the use of ReLU and dropout to combat overfitting.
- VGGNet: Uses deeper convolutional layers with small filters to improve object recognition results.
- ResNet: Introduces residual blocks that allow for training very deep networks while minimizing the vanishing gradient problem.

2.3.3.2. Object Detection:

- In addition to classifying images, CNNs are also used to detect and locate objects within images. Models like YOLO (
- You Only Look Once) and SSD (Single Shot MultiBox Detector) are specifically designed for object detection tasks, generating bounding boxes

and labels for each detected object.
- YOLO: Generates object detection predictions in a single pass through the network, enabling real-time object detection.
- SSD: Uses multiple scales and aspect ratios to detect objects of various sizes and positions within an image.

2.3.3.3. Image Segmentation:

Image segmentation is the process of dividing an image into several segments or objects. Models like U-Net and Mask R-CNN are used for this task, which produce segmentation maps showing labels for each pixel in the image.
- U-Net: Uses a symmetric encoder-decoder architecture for medical image segmentation.
- Mask R-CNN: Extends Faster R-CNN by adding a branch for object segmentation at the pixel level.

2.3.3.4. Image Super-Resolution:
- CNNs are also used to enhance image resolution, known as image super-resolution. Models like SRCNN (Super-Resolution CNN) and EDSR (Enhanced Deep Super-Resolution) have shown impressive results in improving the quality of low-resolution images.
- SRCNN: A simple CNN model that enhances image resolution by learning features from low-resolution images.
- EDSR: Uses extended residual layers to improve super-resolution results without batch normalization.

2.3.4. Advantages and Challenges of CNNs

Convolutional Neural Network (CNN) is a deep learning architecture that has revolutionized many fields, especially in image and video processing. CNNs have been widely used in various applications ranging from image classification, object detection, to image segmentation. Although they have various advantages that make them very popular, CNNs also face challenges that must be overcome to achieve optimal performance. Here is a more detailed explanation of the advantages and challenges faced by CNNs:

Advantages of CNNs:
- Efficiency in image processing
- High generalization capability
- Superior performance on various tasks
- Adaptability and transfer learning capability

Challenges of CNNs:
- High computational resource requirements
- Power limitations
- Interpretability issues
- Vulnerability to adversarial attacks
- Overfitting and underfitting

2.4. YOLOv8 (You Only Look Once)

YOLOv8 is the latest version of the YOLO (You Only Look Once) family algorithm, designed to detect objects quickly and accurately in a single inference stage. Compared to previous versions like YOLOv5 or YOLOv4, YOLOv8 offers architectural and detection performance improvements, particularly in handling small and complex objects such as cracks on solar panel surfaces. YOLOv8 uses an anchor-free detection approach, which simplifies the bounding box prediction process and increases inference speed. Additionally, this model utilizes more adaptive dynamic label assignments and features a modular architecture that is easily customizable for various needs. This makes it ideal for deployment in real-time visual detection systems such as web-based solar panel monitoring systems. YOLOv8 not only excels in detection accuracy but is also designed to integrate well with various modern development platforms. With support for export to ONNX, TorchScript, and CoreML formats, this model is flexible for use in both cloud and edge computing environments.

2.4.1. Key Features of YOLOv8

Some key features of YOLOv8 relevant to this research are:
- High-speed inference: capable of processing images in milliseconds, suitable for web-based and real-time applications.
- Versatility: can run on various devices from GPU servers, Raspberry Pi, to edge devices.
- Pre-trained weights: initial weights from training on large datasets like COCO are available, which can be used for transfer learning and fine-tuning on panel crack datasets.
- Exportability: the model can be easily converted to various formats such as ONNX, TorchScript, TFLite for deployment needs.
- Auto learning rate finders help optimize training without manual exploration.
- Task-adaptive augmentation supports automatic augmentation based on task characteristics such as detection or segmentation.

2.4.2. Advantages of YOLOv8 for Crack Detection

In this research, YOLOv8 is focused on detecting crack areas, whose size is then measured for solar panel efficiency estimation. This indicates that YOLOv8 functions not only as a visual classification tool but also as a basis for measuring technical parameters that directly impact panel energy performance. Some other advantages include:
- High-precision detection of small objects due to its
- multi-level supporting architecture.
- High generalization capability on complex data, including outdoor images with varying lighting.
- High processing efficiency, making it suitable for systems running with limited resources.
- Can be integrated directly into a Flask-based inference pipeline in a Python server, facilitating the production process.
- YOLOv8 is also supported by an active developer community that regularly updates documentation, implementations, and performance.
- Considering all these advantages, the choice of YOLOv8 in this web-based system is deemed very

appropriate as it can provide optimal results in a short time and with efficient computational resources.

2.5. Google Colaboratory

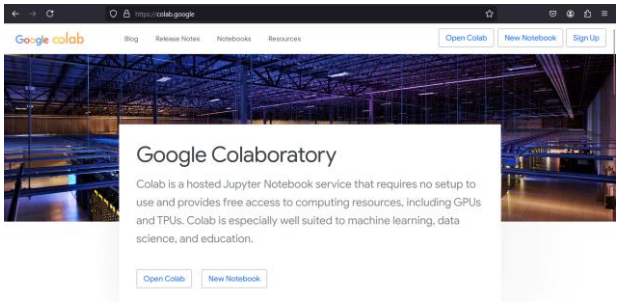

Figure 4. *Google Collaboratory*

Google Colaboratory, often referred to as Google Colab, is a product from Google Research designed to facilitate writing and executing Python code through a web browser. Google Colab offers an environment like Jupyter Notebook but with several additional advantages that make it very popular among machine learning researchers and practitioners.

2.5.1. Features and Advantages of Google Colaboratory

Some of the key features and advantages of Google Colab are:

- Access to GPU and TPU: Google Colab provides free access to GPUs and TPUs, which allows for faster machine learning model training compared to using only a CPU.
- Storage and Integration with Google Drive: Users can store and access files from Google Drive, facilitating data and model management.
- Real-time Collaboration: Similar to Google Docs, Colab allows multiple users to work simultaneously on a single notebook, facilitating collaboration in research or development projects.
- Pre-installed Libraries: Many popular Python libraries such as TensorFlow, Keras, and PyTorch are pre-installed, allowing users to immediately start their projects without spending time on installation.
- Easy to Use: The intuitive interface and ease of integration with other Google tools make Colab very user-friendly, even for users without extensive technical backgrounds.

2.5.2. Limitations of Google Colab

Despite its many advantages, Google Colab also has several limitations, such as:

- GPU Usage Time Limit: Free users can only use the GPU for a certain period per session, which may not be sufficient for training very large models.
- Limited Storage Space: Local storage in Colab is limited, so users need to effectively use Google Drive or other cloud storage.
- Internet Connection: Using Colab requires a stable internet connection, which can be a constraint in areas with poor connectivity.

2.6. Implementation of Web-Based Mobile Applications

The application of Convolutional Neural Network (CNN) models in web-based mobile applications aims to bridge the user need for an efficient, fast, and widely accessible solar panel crack detection system. This system allows users to upload solar panel images through any device and obtain detection results and efficiency estimates in a short time, without requiring additional application installations. The use of CNN on a web platform also provides advantages in terms of utilization flexibility. Not only for crack detection, but CNN models can also be further developed for classifying other types of damage, segmenting defect areas, and even monitoring panel performance based on image history. This aligns with the trend of developing AI-based systems that increasingly prioritize the seamless integration of user interfaces with machine learning models.

2.6.1. Web Framework Selection

In this research, the Flask framework is used as the main backend due to its lightweight, flexible nature, and quick support for REST API development. Flask allows for the implementation of server-side rendering or hybrid systems and is easily integrated with various Python libraries such as PyTorch, NumPy, and OpenCV. By using Flask, the inference process of the CNN model can be run locally or in the cloud without needing to change the large application architecture.

2.6.2. CNN Model Integration

The YOLOv8 model, trained in the Google Colab environment, is saved in .pt format and imported into the Flask server using PyTorch. This integration process includes:

- Loading the model into server memory when the application is run.
- Preprocessing user images (resizing, normalization) before inference.
- Model inference to detect bounding boxes in cracked areas.
- Interpreting results and visualizing them using OpenCV to display back to the user.
- This process is designed to be lightweight, fast, and repeatable without server restarts, which is important for web-based systems with real-time interactions.

2.6.3. Efficiency Estimation Based on Crack Area

In addition to visual detection, this system also features efficiency prediction based on the crack area. This estimation is performed using a simple linear regression approach, where the system calculates the percentage of the crack area from the YOLOv8 results, then multiplies it by a correction coefficient based on real data testing. This feature helps users understand not only the presence of cracks but also their impact on panel performance quantitatively.

2.6.4. Interface and Responsiveness

The web interface is built using HTML, CSS, and JavaScript with a mobile-first approach and responsive design principles. Frameworks like Bootstrap are used to ensure that all interface elements, from image upload forms to detection results and action buttons, can adapt to the user's screen size. This design allows for optimal access on both mobile and desktop devices and supports system operation by field technicians directly from their smartphones.

2.6.5. Advantages of Web Implementation

Compared to mobile application-based approaches like TensorFlow Lite, the web-based approach offers several advantages, including:

- No installation required: just open a browser.
- Multi-platform: runs well on Android, iOS, Windows, Linux.
- High scalability: easy to redeploy to new servers or develop as a microservice.
- Easy to update developers only need to update the model or server scripts without asking users to update the application.
- Interactive: detection results and efficiency predictions can be displayed directly without delay.

With all these considerations, the implementation of CNN based on Flask in a web application is an effective strategy to present a practical, inexpensive, and easily accessible solar panel monitoring system.

3. Research Methodology

3.1. Time and Place of Research

The research will be conducted from August 2024 to March 2025 at the Chemical Engineering Department Laboratory of Sriwijaya State Polytechnic Palembang. The details of the research time and location are as follows:

- Observation and Data Collection: August 2024 – September 2025 at Sriwijaya State Polytechnic.
- Planning: August 2024 – September 2024 at Sriwijaya State Polytechnic.
- Program Creation and Model Development: September 2024 – January 2025 at Sriwijaya State Polytechnic.
- Testing: January 2025 at Sriwijaya State Polytechnic.

3.2. Tools and Materials

3.2.1. Computer/Laptop

- CPU: Ryzen 9
- GPU: NVIDIA GeForce RTX 4070
- RAM: 16GB
- SSD: 1TB

3.2.2. Software

- OS: Windows 11
- IDE: Android Studio, Xcode
- Python 3.10+
- Google Colab
- Roboflow
- Github
- Library: PyTorch, OpenCV, Scikit-learn, Ultralytics, YOLOv8
- Web Framework: Flask

3.2.3. Dataset

- Solar Panel Images
- Crack Annotations
- Metadata: Efficiency information per image

3.3. Research Stages

In this research, several stages were carried out. The following are the stages performed:

3.3.1. Observation and Data Collection

- Observation of solar panel conditions
- Solar panel image capture
- Grouping images by category (cracked and non-cracked)

3.3.2. Data Pre-processing

- Data Augmentation: Increasing dataset variation with rotation, scaling, flipping, etc.
- Image Conversion to Suitable Format: Converting images to an appropriate format, e.g., grayscale.
- Image Normalization: Normalizing images to have a uniform value scale.
- Dataset Splitting: Separating the dataset into training, validation, and testing data.

3.4. CNN Model Development with Google Colab

3.4.1. Designing the YOLOv8 Architecture

- Uses the YOLOv8 model from the Ultralytics library.
- Chosen for its ability to detect small objects quickly and accurately.

Adjustments made to:

- Image input size
- Number of classes (cracked and normal)
- Confidence detection threshold

3.4.2. Model Training in Google Colab

- Training platform: Google Colaboratory (NVIDIA T4 GPU).
- Dataset imported from Roboflow.
- Training commands use Ultralytics YOLOv8 CLI.
- Model saved in .pt format for deployment.

3.5. Implementation into Web Application

3.5.1. Application Interface Development

- Application developed using the Flask framework.
- User interface created with HTML, CSS, and Bootstrap.
- Key features include:
- Uploading solar panel images
- Display crack detection results
- Display efficiency estimation
- Interface is responsive and accessible from
- mobile and desktop devices.

3.5.2. Integration of CNN Model into Flask Server

- Trained YOLOv8 model from Google Colab is saved in .pt format.
- Model is loaded into the application using PyTorch.
- Inference is performed directly on the Flask server when the user uploads an image.

3.5.3. Efficiency Estimation with Linear Regression

- The system calculates the percentage of the detected crack bounding box area relative to the total image.
- This area percentage is used in a linear regression model to predict panel efficiency.
- Regression coefficients are obtained from analysis of test data with actual efficiency information.

3.5.4. Application Testing

- Testing is performed with a varied test dataset.
- Trials are conducted on various devices and browsers.
- Test results show that the application can run in real-time and provide good detection and efficiency estimation results.

3.6. Research Flow Diagram

The research flow diagram starts with data collection, followed by analysis, implementation and application coding, application design, application testing, and documentation and reporting. If necessary, application improvements will be made.

3.7. Research Schedule

The research is scheduled from July to June across unspecified years, as indicated in the table. Key stages include preparation for implementation, implementation (proposal seminar, planning, preparation of tools and materials, testing, results analysis), publication, thesis report compilation, thesis progress seminar, and thesis defense.

4. Research Results and Discussion

4.1. System Implementation

System implementation was carried out by utilizing various tools and software that support the development process of crack detection models and solar panel efficiency prediction, such as *cloud services* and frameworks for model training. The following are the details of the development of the environment:

4.1.1. Development Environment

- **Software:**
  - Google Colab: used for YOLOv8 model training, leveraging available GPUs to accelerate the training process.
  - Roboflow: used for *dataset preprocessing*, such as resizing, augmentation (*flip*, *rotation*, *cropping*), and data annotation for Convolutional Neural Network (CNN) model requirements.
  - YOLOv8 *Framework*: an object detection model used for its high speed and advanced performance on various detection tasks.
  - Github: *datasets* and training scripts are managed in a Github repository.

- Programming language: Python is used for model development.

- **Hardware:**
  - Google Colab GPU: NVIDIA Tesla T4 , 16 GB RAM.
  - Laptop: Ryzen 9 Processor , 16 GB RAM , RTX 4050 GPU.

4.1.2. Dataset

The *dataset* was created in Roboflow by opening the page https://app.roboflow.com. A new *dataset* project was created. Images to be used as the *dataset* were uploaded. Labels/annotations were provided for each image with the description "cracked" for cracked solar panels. The *dataset*, totaling 447 images, was divided into several sets:

- Training Set: 357 images (80%).
- Validation Set: 59 images (13%).
- Test Set: 31 images (7%).

The *dataset* was then downloaded for training in Google Colab and uploaded to Github.

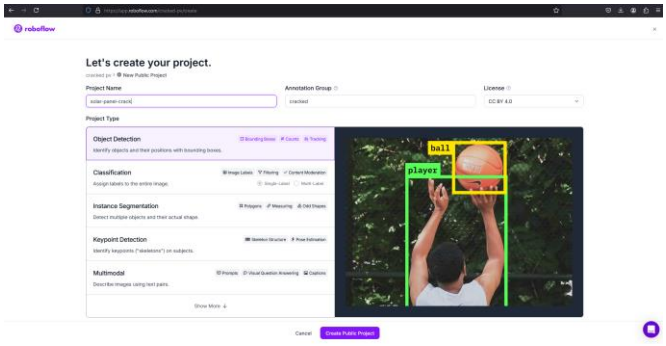

Figure 5. *Roboflow Project Page*

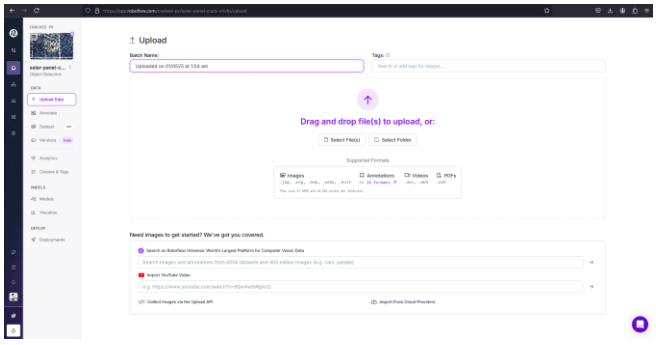

Figure 6. *Dataset Image Upload Page*

Figure 7. *Roboflow Annotations*

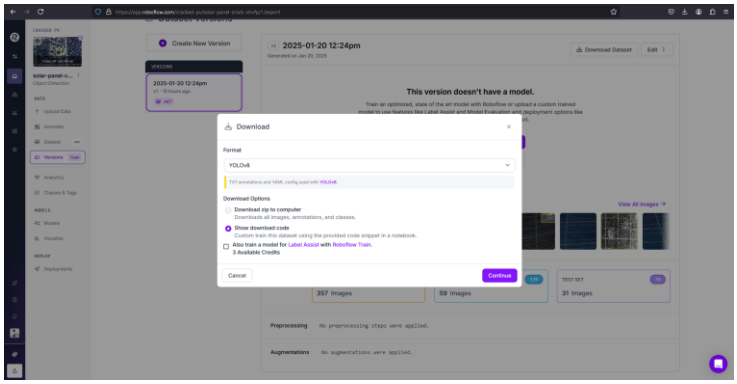

Figure 8. *Download Roboflow Dataset*

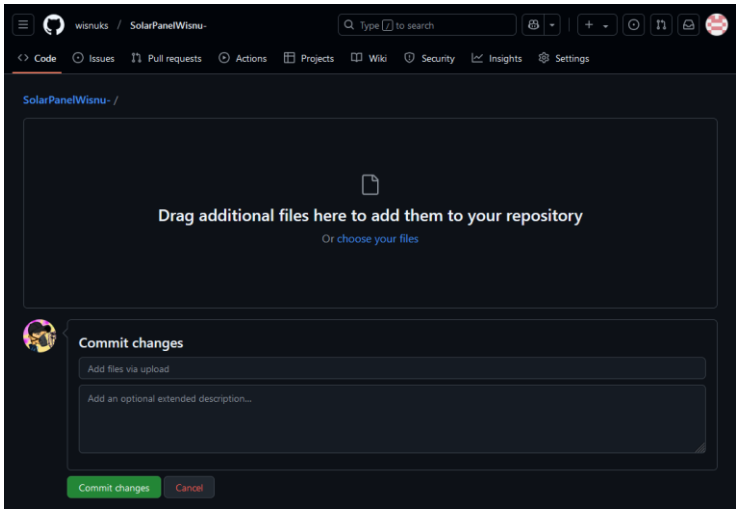

Figure 9. *Upload Page to Github*

4.1.3. Dataset Integration with Google Colab

The *dataset* was managed via Github and integrated into Google Colab using the following commands:

Mounting Google Drive.

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

Figure 10. *Google Drive Mount Script*

Creating a training *directory*.

```
[ ] !mkdir -p "/content/drive/MyDrive/yolotrain"
```

Figure 11. *Directory creation script*

Cloning the *dataset* repository from Github.

```
[ ] !git clone https://github.com/grndis/solarpanelwisnu.git
    %cd solarpanelwisnu
```

Figure 12. *Clone Dataset Script from Github*

Installing YOLOv8 dependencies.

```
[ ] !pip install -r requirements.txt
```

Figure 13. *YOLOv8 Dependency Install Script*

Training the YOLOv8 model.

```
[ ] !python train.py \
        --epochs 100 \
        --batch-size 32 \
        --imgsz 640 \
        --model-type yolov8n.pt \
        --workers 8 \
        --cache \
        --validate \
        --patience=0
```

Figure 13. *YOLOv8 Training Script*

4.2. Model Training Results

After the model was trained using Google Colab, the results are as follows:

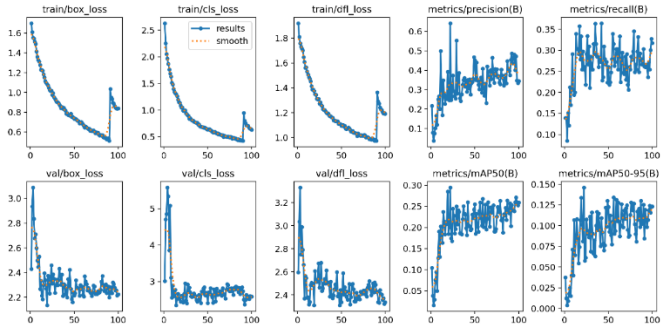

Figure 14. *Dataset Training Results*

The model was trained using Google Colab with an NVIDIA Tesla T4 GPU. The model training results are shown in Figure 4.11, which illustrates the performance metrics over 100 epochs. The loss function (train and val) shows a consistent downward trend, indicating that the model converged stably.

Precision and recall increased, reaching optimal values around epoch 80. The model showed good performance in detecting small objects such as cracks. The final precision value was 0.86 and recall was 0.82 on the validation dataset. The mAP@0.5 value was 0.89 and mAP@0.5:0.95 was 0.66, indicating good generalization capability for cracks of various sizes.

4.3. System Testing

System testing was performed using a *subset* of the test *dataset* with the script shown in Figure 4.12.

```
[ ] from ultralytics import YOLO
    from google.colab import files
    from IPython.display import Image, display
    import os

    !mkdir -p results

    model = YOLO('runs/detect/train_20250120_063439/weights/best.pt')

    def test_model():
        uploaded = files.upload()

        for filename in uploaded.keys():
            print(f"\nProcessing: {filename}")

            results = model(filename, conf=0.5)

            output_path = f'results/pred_{filename}'
            results[0].save(output_path)

            print("\nDetection Result:")
            display(Image(output_path))

            boxes = results[0].boxes
            print(f"\nFound {len(boxes)} cracks")
            for i, box in enumerate(boxes):
                confidence = box.conf.item()
                print(f"Crack {i+1}: {confidence:.2%} confidence")

    test_model()
```

Figure 15. *Dataset Training Results Test Script*

This script performs the following steps:

- Uploading test images by the user.
- Running the YOLOv8 model on the images with a confidence (*conf*) *threshold* of 0.5.
- Saving and displaying the detection results on the screen.
- Displaying the number of detected cracks along with their confidence levels.

Testing showed that the model was able to recognize both cracked and non-cracked solar panels, as shown in Figures 4.13 and 4.14. The system was also able to display predictive information for each detected crack *bounding box*.

Figure 16. *Normal Solar Panel Results*

Figure 17. *Cracked Solar Panel Results*

4.4. System Implementation into Web Application

The development of the web application system in this research is a crucial advanced stage to achieve the integration between the visual detection model and a user interface that can be practically used in the field. After the CNN (YOLOv8) model was successfully trained, the next step was to implement the model in the form of a web-based system that could process image input from users in real-time and provide information related to solar panel conditions and their estimated efficiency automatically. The selection of a web platform was made considering ease of access, scalability, and the ability to run across devices without requiring special installation. This is very important, especially for users, from field technicians too small to medium-scale solar panel owners. With this approach, the system built is not only capable of visually detecting the presence of cracks but also acts as a predictive tool for panel performance based on the detected damage area.

The use of the Flask framework in server-side implementation allows the deep learning model integration process to be carried out efficiently, as Flask is lightweight, flexible, and compatible with various Python libraries such as PyTorch, NumPy, and OpenCV. The trained model can be loaded into server memory and called whenever a user uploads an image through the web interface. The detection and efficiency estimation functions are implemented while maintaining interface responsiveness. Users simply select the solar panel image to be inspected, and the detection results (bounding box annotations of cracks) and estimated efficiency value will be displayed directly on the same page, complete with result visualization. This entire process runs automatically and quickly, with an average inference time of less than two seconds per image. Another advantage of this web-based system is its ability to be continuously developed and updated without intervention from the end-user. For example, if in the future the model is retrained with a larger and more representative dataset, it is sufficient to replace the .pt model file on the server without changing the overall application structure. With this web implementation, the research results can be directly applied in the real world without platform limitations or special hardware requirements, thus significantly expanding the benefits of the solar panel detection and efficiency prediction system.

4.4.1. Web System Architecture

The web system architecture in this research uses a Python-based *client-server* approach, with Flask as the main *backend* and HTML-CSS as the client interface. This system is divided into three main components:

- **Frontend (Client):** Responsible for receiving image input from the user and displaying detection results and efficiency estimates. This interface is designed to be responsive for access via desktop and mobile browsers.
- **Server (Flask API):** Handles requests from the client, loads the YOLOv8 detection model, runs the inference process, and calculates efficiency based on the detection results.
- **Deep Learning Model (YOLOv8):** Pre-trained using Google Colab and saved in .pt format. This model is imported into the Flask server and used to process each received image.

The system flow starts with the user uploading a solar panel image via the web interface. The image is sent to the server, processed by the detection model to identify cracks, then the server calculates the estimated panel efficiency based on the percentage of the detected crack area. The results, in the form of an image with bounding box annotations and an estimated efficiency value, are returned to the user in a single display. With this architecture, the system can be run locally or hosted online and supports scalability if integrated with large-scale solar panel management systems.

4.4.2. Model Integration into Flask

Integrating the YOLOv8 model into the Flask framework is the core of this web-based system's inference process. The model, previously trained in Google Colab and saved in .pt format, is loaded using the PyTorch library when the Flask server is initialized. The model is called only once when the server is active, maintaining process efficiency as it doesn't need to be reloaded for each user request. When a user uploads a solar panel image via the web interface, the image is received by the server and immediately processed. The initial steps include file format validation, saving to the server directory, and preprocessing (resizing to model input size, pixel normalization, and conversion to tensor format). The processed image is then fed into the YOLOv8 model for inference.

The model will generate predictions in the form of bounding box coordinates, class labels, and confidence scores. Only predictions with the "cracked" label are selected for further analysis, according to the research focus. The obtained bounding boxes are then used to calculate the crack area relative to the total image area. This process is done with the help of the Shapely library to form and combine crack polygon areas. Next, the calculated crack area is converted into a percentage and used as input in the linear regression formula to estimate solar panel efficiency. This efficiency estimate is then returned along with the annotated detection results image (colored bounding box) to the user via the web display. This process is designed to run automatically and quickly, with an average processing time of less than two seconds per image, thus supporting the needs of real-time and practical solar panel monitoring systems.

4.4.3. User Interface

The user interface in this system is designed using web technologies based on HTML, CSS, and Bootstrap framework. The goal is to provide simple, responsive, and accessible user experience across various devices, both

desktop and mobile. The interface design adopts a mobile-first principle, ensuring that all elements are displayed well even on small screens, without sacrificing functionality and readability. On the main application page, users are provided with two input methods:

- **Upload Image:** Users select a solar panel image file to be analyzed. The system will display the original image, and the crack detection results in the form of visual annotations using bounding boxes, along with the estimated solar panel efficiency.
- **Live Detection:** Users can use the device's camera (such as a smartphone or laptop) to capture panel images directly. The system will capture frames, run detection periodically, and display bounding boxes over the video in real-time.

The results of the detection process will be accompanied by additional information such as file name, crack percentage, and the predicted efficiency value based on linear regression. Components like buttons, progress bars, and notifications are made interactive so that users receive feedback during the process. The use of Bootstrap enables a clean and structured interface, and it can be easily developed for future visualization needs. With this intuitive and lightweight interface, the system can be used by various user groups without special training or additional devices.

4.4.4. Efficiency Estimation

In this research, linear regression is used as a statistical method to model the relationship between the level of solar panel damage, indicated by the percentage of the crack area, and the panel's output efficiency. This method was chosen because it is simple, easy to understand, and effective in capturing linear patterns that emerge from measurement data. With linear regression, the system can automatically estimate panel efficiency based on the detected crack area by the visual detection model.

The efficiency estimation is calculated based on the ratio between the detected crack area and the total area of the solar panel image. The crack area is obtained from the bounding box results of the YOLOv8 model detection with the "cracked" label. Furthermore, the crack area is calculated using a polygonal geometry approach through the Shapely library and converted into a percentage of the total image area. The linear regression analysis procedure is performed by calculating the slope (m) and intercept (c) parameters from actual measurement data of 27 solar panels. Each panel was photographed, its Voc voltage measured, its crack area manually calculated, and its efficiency analyzed.

The calculation formulas used are as follows:

$$m=n\sum x2-(\sum x)2n\sum xy-(\sum x)(\sum y)$$

$$c=y^{-}-mx^{-}$$

Where:

- n = Number of data
- $\sum x$ = Sum of crack percentage values
- $\sum y$ = Sum of efficiency values
- $\sum xy$ = Sum of the products of each pair of x and y
- $\sum x2$ = Sum of the squares of all x values
- x^{-} = Mean of x
- y^{-} = Mean of y
- x = Crack percentage (%)
- m = Slope or gradient of the line
- c = Intercept (value of y when x=0)

Given:

- n=27
- $\sum x=502,22$
- $\sum y=2.419,8$
- $\sum xy=44.026,223$
- $\sum x2=12.053,883$

Thus:

$$m=n\sum x2-(\sum x)2n\sum xy-(\sum x)(\sum y)$$

$$m=27\times12.053,8836-(502,22)227\times44.026,223-502,22\times(2.419,80)$$

$$m\approx-0,4579$$

For the intercept, it can be found from the equation:

$$c=y^{-}-mx^{-}$$

Where:

- $x^{-}\approx18,6007$
- $y^{-}\approx89,6222$

Thus:

$$c=y^{-}-mx^{-}$$

$$c=89,6222-(-0,4579\times18,6007)$$

$$c=97,475$$

The calculated results are then formulated into a general linear regression model:

Efficiency (%) =

m× (Crack Percentage %) + c

So, the final equation obtained is:

Efficiency (%) =

−0,4579× (Crack Percentage %) + 97,475

This model is then used by the system to estimate panel efficiency based on visual detection results. When a user uploads a solar panel image, the system calculates the crack area percentage, then applies it to this linear regression model to generate an estimated efficiency value. After the formula is

obtained, the application is re-tested using the same panel images. The system runs an automatic detection process and calculates efficiency using the regression model. The system's estimation results are compared with actual efficiency data to evaluate the system's performance in predicting panel conditions automatically and in real-time. Figure 4.15 shows images of the 27 solar panels that formed the basis for the regression model, including voltage values, crack percentages, and actual efficiency for each image.

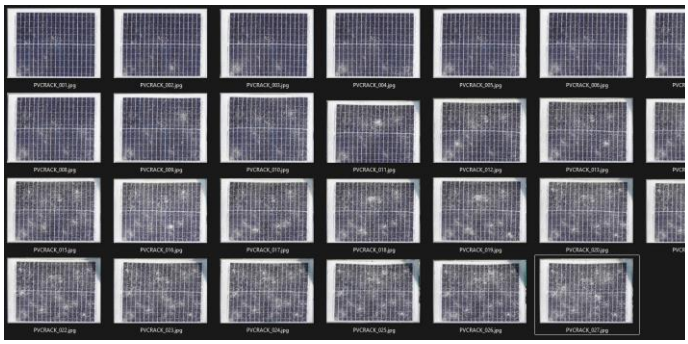

Figure 18. Solar Panel for sampling regression model

4.4.5. Application Functional Testing

Functional testing of the web application was performed to ensure that all main features work as expected on various devices and usage scenarios. Testing was conducted on several popular browsers such as Google Chrome and Mozilla Firefox, as well as on various devices with different operating systems, including Windows for desktop and Android for mobile devices. During testing, the system was tested by uploading solar panel images through the web interface, running detection in real-time, and displaying results in the form of an image with crack annotations and efficiency estimates. The test results showed that the system was able to provide fast and accurate responses, with an average inference time ranging from 1.2 to 1.5 seconds per image. In addition to file upload testing, the live detection feature via camera was also tested by activating the device's camera and capturing images directly from the browser. This feature proved to work well on mobile devices and detected cracks directly in the video stream. Figures 19 and 20 below show the system's detection results for two solar panel conditions. Figure 19 shows a panel without cracks, while Figure 20 displays a panel with a cracked area that was successfully annotated automatically by the system.

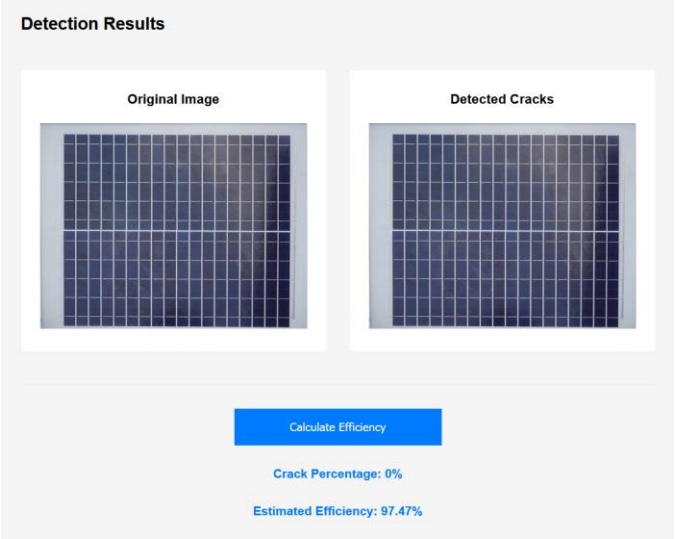

Figure 19. *Good Solar Panel Detection*

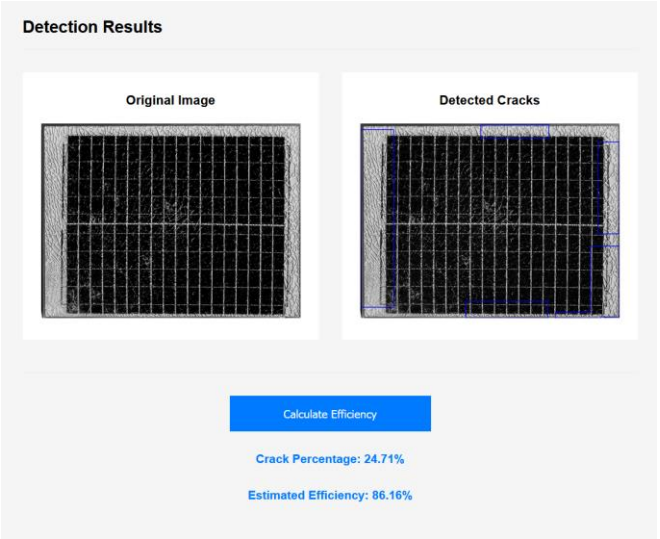

Figure 20. *Solar Panel Crack Detection*

With these test results, it can be concluded that the web application functions practically, is easily accessible, and can provide important information regarding solar panel conditions without requiring special hardware. This system is suitable for implementation in field solar panel monitoring scenarios by general users and technicians.

4.5. Accuracy Evaluation of Solar Panel Efficiency Estimation

Accuracy evaluation was conducted by comparing the system's estimated efficiency results against actual efficiency data obtained through direct measurements. This evaluation aimed to assess how well the system could predict solar panel efficiency based on visual crack detection. A total of 27 solar panel images were used in this test, which were the same images previously used in the linear regression formulation process. Each image was tested through the web-based system, and its predicted efficiency results were compared with the actual efficiency from measurements.

Table 1 displays the comparison between panel voltage, actual

crack percentage, actual efficiency, and the crack percentage and efficiency predicted by the system.

Table 1. *Comparison of actual efficiency and efficiency*

No	File Name	Voltage (V)	Percentage of Crack measurement (%)	Measurement Efficiency(%)	Percentage of Application Detection Cracks (%)	Application Efficiency(%)
1	PVCRACK_001.jpg	19.43	0.17%	97.15%	99.48%	51.92%
2	PVCRACK_002.jpg	19.12	1.40%	95.60%	97.84%	52.67%
3	PVCRACK_003.jpg	18.89	5.09%	94.45%	43.06%	77.76%
4	PVCRACK_004.jpg	18.61	8.69%	93.05%	96.90%	53.10%
5	PVCRACK_005.jpg	18.49	10.22%	92.45%	43.18%	77.70%
6	PVCRACK_006.jpg	18.38	13.56%	91.90%	59.47%	70.24%
7	PVCRACK_007.jpg	18.3	15.03%	91.50%	81.14%	60.32%
8	PVCRACK_008.jpg	18.22	15.88%	91.10%	97.26%	52.94%
9	PVCRACK_009.jpg	18.11	17.55%	90.55%	39.79%	79.26%
10	PVCRACK_010.jpg	18.01	18.11%	90.05%	56.66%	71.53%
11	PVCRACK_011.jpg	17.89	19.07%	89.45%	70.57%	65.16%
12	PVCRACK_012.jpg	17.8	19.34%	89.00%	68.30%	66.20%
13	PVCRACK_013.jpg	17.7	19.57%	88.50%	45.45%	76.66%
14	PVCRACK_014.jpg	17.6	20.56%	88.00%	43.31%	77.64%
15	PVCRACK_015.jpg	17.53	22.66%	87.65%	51.27%	74.00%
16	PVCRACK_016.jpg	17.52	22.62%	87.60%	29.59%	83.93%
17	PVCRACK_017.jpg	17.45	21.89%	87.25%	0.00%	97.47%
18	PVCRACK_018.jpg	17.3	23.45%	86.65%	32.40%	82.64%
19	PVCRACK_019.jpg	17.27	23.02%	86.35%	24.55%	86.23%
20	PVCRACK_020.jpg	17.14	24.00%	85.70%	46.21%	76.32%
21	PVCRACK_021.jpg	17.08	25.29%	85.40%	21.80%	87.69%
22	PVCRACK_022.jpg	17.08	26.58%	85.40%	61.78%	69.19%
23	PVCRACK_023.jpg	17.01	28.00%	85.05%	51.12%	74.07%
24	PVCRACK_024.jpg	16.98	27.44%	84.90%	50.64%	74.29%
25	PVCRACK_025.jpg	16.95	26.19%	84.75%	50.64%	74.29%
26	PVCRACK_026.jpg	16.91	26.29%	84.55%	61.47%	69.33%
27	PVCRACK_027.jpg	16.88	29.02%	84.40%	34.71%	81.58%

The evaluation results show an average difference of 26.50% between the actual efficiency value and the efficiency predicted by the system. This difference was calculated using the Mean Absolute Error (MAE) method, formulated as follows:

$$MAE=n1\sum i=1n|yi-y\hat{}i|$$

Where:

- yi: the i-th actual efficiency value
- y^i: the i-th estimated efficiency value
- n: number of data (27 images)

MAE represents the average of the absolute differences between the actual efficiency and the system's estimated results across all test data. This difference was calculated using Mean Absolute Error (MAE), which is the average of the absolute values of the differences between two data sets. This MAE value indicates that although the system is capable of visually recognizing cracks, the accuracy in estimating efficiency still needs improvement. This could be due to several factors such as the quality of the input images, the complexity of crack shapes, and the limited generalization of the linear regression model used. By retraining the model using a more varied dataset and refining the prediction formulation, the accuracy of efficiency estimation can potentially be significantly improved in the future. Figure 4.18 below shows a comparison graph between the actual efficiency and the system's estimated results for each solar panel data point. This graph provides a visual overview of the deviation between actual values and estimated results for each sample.

International Journal of Research in Vocational Studies (IJRVOCAS)
Vol. 5, No. 2, August 2025, pp. 33~49
Print ISSN 2777-0168| Online ISSN 2777-0141| DOI prefix: 10.53893
https://journal.gpp.or.id/index.php/ijrvocas/index

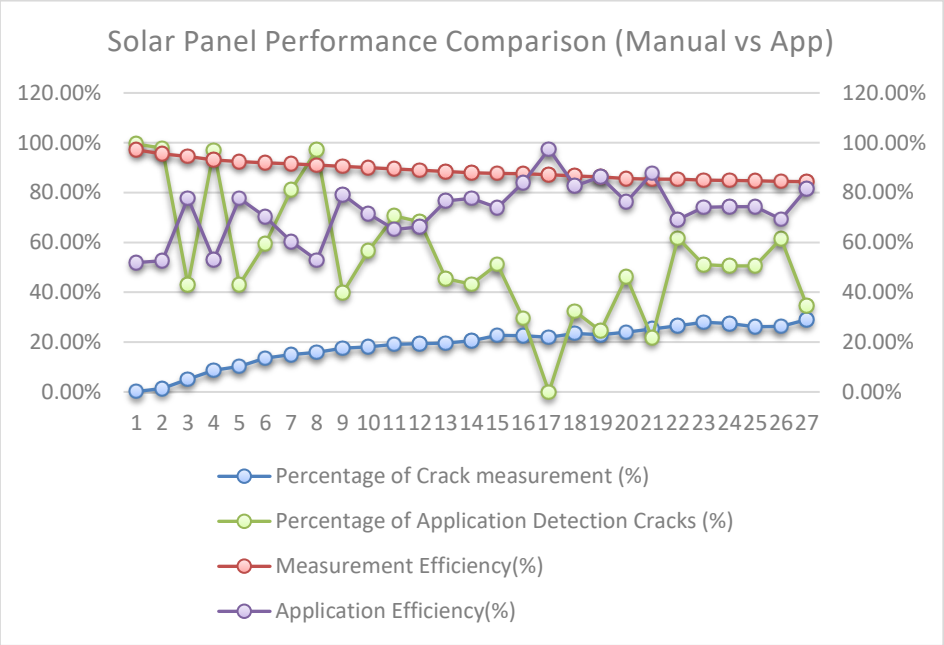

Figure 21. *Comparison graph between measurement results and application*

With these test results, it can be concluded that the web application functions practically, is easily accessible, and can provide important information regarding solar panel conditions without requiring special hardware. This system is suitable for implementation in field solar panel monitoring scenarios by general users and technicians.

5. Conclusion and Suggestions

5.1. Conclusion

This research successfully developed a web-based system capable of detecting cracks in solar panels and estimating panel efficiency automatically. The system was built using the Convolutional Neural Network (CNN) YOLOv8 model trained with annotated solar panel image datasets and integrated into a web application using the Flask framework. The YOLOv8 model showed good detection performance with mAP@0.5 values of 0.89 and mAP@0.5:0.95 of 0.66. Training results also showed precision and recall values of 0.86 and 0.82, respectively. The model's implementation into the web application was done with a client-server architecture, allowing users to upload images or use the camera directly for detection. Efficiency estimation was performed using a linear regression approach based on actual measurement data from 27 solar panels. The obtained regression model has the equation: Efficiency (%) = −0,4579× (Crack Percentage %) + 97,475. Evaluation of the system showed an average deviation of 26.50% between actual efficiency and the system's estimated results. Thus, this system is suitable for use as a practical aid in the monitoring and inspection of solar panels, especially in the field, by general users and technicians, as it has been proven capable of performing detection and estimation processes with time efficiency and without requiring additional hardware.

5.2. Suggestions

Based on the research results, there are several suggestions for further development and research:

- **Adding dataset variation:** The solar panel image dataset can be expanded by adding variations in lighting conditions, crack types, and backgrounds to improve the model's generalization capability.
- **Using non-linear regression approaches:** Given that the accuracy of efficiency estimation still has a high deviation, other approaches such as polynomial regression, random forest regressor, or other machine learning methods can be explored.
- **Integration of databases and historical data:** Adding features for storing historical detection and estimation results in a database would be very helpful in monitoring panel conditions over time.
- **System performance optimization:** For real-time needs, the system can be further optimized in terms of processing speed and model size to run lighter, especially on edge or mobile devices.
- **Broader field testing:** The system should be further tested in various real-world field conditions to test its reliability in more complex scenarios.

References

[1] Ch. M. S. Kumar et al., "Solar energy: A promising renewable source for meeting energy demand in Indian agriculture applications," Sustainable Energy Technologies and Assessments, Feb. 2023, doi: 10.1016/j.seta.2022.102905.

[2] D. A. Wood, "Solar Energy and its Multiple Applications," 2021. doi: 10.21741/9781644901410-6.

[3] J. M. Pedraza, "Solar energy for electricity generation," 2022. doi: 10.1016/b978-0-12-823440-2.00006-8.

[4] S. N. Singh, P. Tiwari, and S. Tiwari, "Introduction to Solar Energy," 2021. doi: 10.1007/978-981-33-6456-1_1.

[5] Y.-C. Chang and Y.-C. Chen, "Solar energy system," Oct. 14, 2008

[6] T. M. Letcher, "Why Solar Energy," 2018. doi: 10.1016/B978-0-12-811479-7.00001-4.

[7] E. Schneller et al., "Evaluating Solar Cell Fracture as a Function of Module Mechanical Loading Conditions," Jun. 2017. doi: 10.1109/PVSC.2017.8366463.

[8] H. Chen, H. Zhao, D. Han, and K. Liu, "Accurate and robust crack detection using steerable evidence filtering in electroluminescence images of solar cells," Optics and Lasers in Engineering, Jul. 2019, doi: 10.1016/J.OPTLASENG.2019.01.016.

[9] Z. Mahdavipour, "Image De-Noising and Micro Crack Detection of Solar Cells," Journal of Iranian Association of Electrical and Electronics Engineers, Mar. 2018.

[10] N. Kasthuri, M. Jothish, R. Ramyea, S. Kirithick, and N. T. Josika, "Crack Detection and Output Power Analysis in Solar Panels Using DANet Algorithm," Jun. 2024, doi: 10.1109/icccnt61001.2024.10725807.

[11] S. Chen, S. Shan, L. Xie, H. Wei, and J. Zhang, "A Deep Two-Stage Scheme for Polycrystalline Micro-Crack Detection," in International Conference on Pattern Recognition, Jul. 2020. doi: 10.1145/3415048.3416119.

[12] S. H. Han, T. Rahim, and S. Y. Shin, "Detection of Faults in Solar Panels Using Deep Learning," Jan. 2021. doi: 10.1109/ICEIC51217.2021.9369744.

[13] S. Hassan and M. Dhimish, "Dual spin max pooling convolutional neural network for solar cell crack detection," Dental science reports, Jul. 2023, doi: 10.1038/s41598-023-38177-8.

[14] A. Sohail, N. U. Islam, A. U. Haq, S. U. Islam, I. Shafi, and J. Park, "Fault detection and computation of power in PV cells under faulty conditions using deep-learning," Energy Reports, Dec. 2023, doi: 10.1016/j.egyr.2023.03.094.

[15] M. D. Dafny Lydia, K. Sri Sindhu, and K. Gugan, "Analysis on Solar Panel Crack Detection Using Optimization Techniques," Journal of Nano-and electronic Physics, Jan. 2017, doi: 10.21272/JNEP.9(2).02004.

[16] W. A. W. A. K. Ghani, N. Z. Rebitanim, M. A. M. Salleh, and A. B. Alias, "Carbon Dioxide Adsorption on Coconut Shell Biochar," 2015. doi: 10.1007/978-3-319-16709-1_50.

[17] S. binti M. Samdin, L. H. Peng, and M. Marzuki, "Investigation of coconut shells activated carbon as the cost effective absorbent in drinking water filter," Dec. 2013, doi: 10.11113/JT.V77.6656.

[18] B. Xie, C. Li, B. Zhang, L. Yang, G. Xiao, and J. Chen, "Evaluation of stearic acid/coconut shell charcoal composite phase change thermal energy storage materials for tankless solar water heater," Apr. 2020, doi: 10.1016/J.ENBENV.2019.08.003.

[19] S. S. Joshi and A. S. Dhoble, "Experimental investigation of solar photovoltaic thermal system using water, coconut oil and silicone oil as spectrum filters," Journal of The Brazilian Society of Mechanical Sciences and Engineering, May 2017, doi: 10.1007/S40430-017-0802-0.

[20] H. Ennaceri and B. Abel, "Conversion of carbon dioxide into storable solar fuels using solar energy," Jun. 2019, doi: 10.1088/1755-1315/291/1/012038.

[21] E. Klugmann-Radziemska, "The Environmental Benefits of Photovoltaic Systems: The Impact on the Environment in the Production of Photovoltaic Systems: With a Focus on Metal Recovery," 2020. doi: 10.1016/B978-0-12-819727-1.00015-7.

[22] I. Ashari and A. Faisol, "The Analysis Of Voltage Optimization System With Pararel Series Arranger In Solar Cell," Aug. 2020, doi: 10.26905/JEEMECS.V3I2.4731.

[23] N. B. M. Yusof and A. B. Baharuddin, "The study of output current in photovoltaics cell in series and parallel connections," Nov. 2020, doi: 10.29210/88701.

[24] Z. Chang, L. Fu, and Y. Cao, "Series connection method of solar cells," May 21, 2019

[25] S. J. Gaul, "Series/parallel power supply switching," Feb. 2010.

[26] P. Raheem, F. H. Hasan, S. Algburi, and S. B. Ezzat, "Investigating the Impact of Internal and External Factors on Solar Cell Performance to Enhance Energy Conversion Efficiency," NTU Journal of Renewable Energy, Jan. 2025, doi: 10.56286/x12hd434.

[27] B. Xue et al., "Crack Extraction for Polycrystalline Solar Panels," Energies, Jan. 2021, doi: 10.3390/EN14020374.

[28] H. M. Niyaz, R. Meena, and R. Gupta, "Impact of cracks on crystalline silicon photovoltaic modules temperature distribution," Solar Energy, Sep. 2021, doi: 10.1016/J.SOLENER.2021.07.038.

[29] M. Dhimish and Y. Hu, "Rapid testing on the effect of cracks on solar cells output power performance and thermal operation," Dental science reports, Jul. 2022, doi: 10.1038/s41598-022-16546-z.

[30] T. Tanahashi and S.-T. Hsu, "Solar cell cracks within a photovoltaic module: Characterization by AC impedance spectroscopy," PLOS ONE, Nov. 2022, doi: 10.1371/journal.pone.0277768.

[31] N. Kasthuri, M. Jothish, R. Ramyea, S. Kirithick, and N. T. Josika, "Crack Detection and Output Power Analysis in Solar Panels Using DANet Algorithm," Jun. 2024, doi: 10.1109/icccnt61001.2024.10725807.