MEDIUM VOLTAGE INSULATOR CRACK DETECTION USING MOBILENetV2 AND TENSORFLOW

Ahmad Yani[1)], Sofyan Sofyan[1)], M.P Lukman[1)], Usman[1)], Apri Junaidi[2)]

[1]Departmet of Electrical Engineering, State Polytechnic of Ujung Pandang, Makassar, 90234, Indonesia.
[2]Faculty of Computing, Universiti Teknologi Malaysia,81310 UTM Johor Bahru, Johor, Malaysia
email: yani150701@gmail.com;sofyantato@poliupg.ac.id;musfirahputrilukman@poliupg.ac.id ; usman@poliupg.ac.id; junaidi20@graduate.utm.my

Abstract

This study discusses the detection of medium voltage insulator cracks using object detection technology. This study uses medium voltage ceramic insulator image data at the ULP Daya Waste Material Warehouse. Ceramic insulator image data is categorized into good and damaged conditions. Preprocessing involves labeling, dividing train data, validation, and testing, and exporting data to Pascal VOC format. MobileNetV2 is implemented on Google Collab to train the object detection model. The evaluation of the model accuracy is in the COCO matrix, while the performance graph shows that the model can read objects well because the reading curve is in line with the smooth curve. Furthermore, this model is applied in creating an Android application that uses the device's camera to detect objects in real-time. This application processes images, converts from YUV to RGB, and performs object detection using the trained model. The detection results are displayed with bounding boxes and labels on the camera reviewer, namely the good class with a reading value of 1.0, the damaged class with a reading value of 0.67 and the background 1.0. This application also tracks detected objects and updates the display according to the detection results.

Keywords: Crack detection, medium voltage insulator, MobileNetV2, Pascal VOC, Android application

I. INTRODUCTION

Maintenance of power system equipment is one of the important tasks [1, 2]. Insulator failure is a crucial problem in many electrical systems, resulting in major operational and safety implications. Insulator failure in gas-insulated switchgear (GIS) has been identified as a significant contributor to GIS failure, representing more than 50% of recorded events from 2013 to 2019 [2]. To reduce insulator failure, it is important to implement preventive measures such as regular cleaning, inspection, and maintenance of insulators and their surrounding areas [3].

Insulator crack inspections carried out by PLN employees so far have been carried out manually by observing directly. The condition of the insulators installed on the medium voltage distribution network at certain locations suspected of having cracks or using drones, the results of the inspections carried out so far have been considered less than optimal because it is difficult to see cracks or defects in medium voltage insulators.

This study discusses the detection of insulator cracks using Artificial Intelligence (AI) technology by focusing on the use of Object Detection technology in medium voltage networks, using the mobileNetV2 and TensorFlow libraries applied to Android smartphone devices[5].

II. LITERATURE REVIEW

A. Isolators in medium voltage distribution networks

Isolators are important components in an electrical power network system that function to separate conductors from other conductors or from the ground [4]. In determining an insulator to be used in an electrical power system, several things need to be noted and

considered, namely, the properties of the material content with the basic material with the ability to withstand bad weather, conditions when contaminated and considerations of production cost issues [5].

Figure 1Material insulator ceramics

B. Python Programming Language

Python was first created by Guido Van Rossum at Scitchting Mathematisch Centrum (CWI) in the Netherlands in the early 1990s. The Python language is motivated by the ABC programming language. Guido is still the main creator of Python, although it is open source, and everyone also participates in its development. In 1995, Guido continued working on Python at the Corporation for National Research Initiative (CNRI) in Virginia, America [6].

C. Image Processing

Image processing is the process of improving an image or picture. Components in image processing include image acquisition sensors and image digitization, computers, storage memory, visualization devices and image printing devices, image processing hardware for specific applications and image processing software [7].

D. Object Detection

Object detection is one of the important tasks in the field of computer vision which aims to identify and determine the location of objects in images or videos [8, 9]. Object detection searches all elements of the image to identify sides that have geometry that matches the desired object in the database. This detection occurs when there is a high enough correlation between the template and the measured area in the image by scanning all positions, scales, and rotations on the template in each image. The use of image-based object recognition has been proven to be accurate and precise against training data [10]. Object detection is useful for recognizing and detecting objects in an image based on color, shape, and from the collected dataset[11].

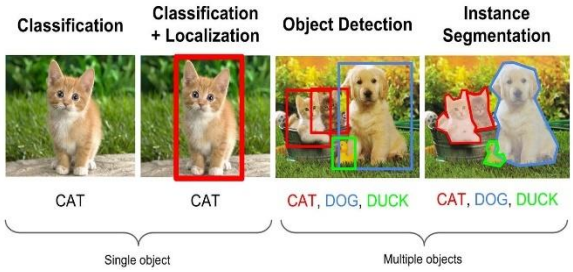


Figure 2Detection steps object

E. MobileNetV2

MobileNetV2 is a neural network architecture created by Google for use with mobile devices or smartphones. MobileNetV2 is the most recent version of the previously developed MobileNet architecture. MobileNetV2 is designed to be small and run quickly on low-end mobile devices. MobileNetV2 employs depth-wise separable convolution, which allows the model to be smaller in size than other CNN models. MobileNetV2 has a number of other features that improve its power efficiency, including the use of bottlenecks and residual layers. MobileNetV2 can also be used on mobile devices to perform real-time tasks like facial recognition, speech recognition, and motion detection [12, 13].

F. TensorFlow

TensorFlow is a computational framework for building machine learning models. TensorFlow provides a variety of toolkits that allow you to create models at your preferred level of abstraction and can run graphs on multiple hardware platforms, including CPUs, GPUs, and TPUs[14].

G. Google Collaboratory

Google Collaboratory or Google Collab is a cloud-based platform for writing, running, and sharing Python code through a web browser. This platform is designed for analysts, developers, researchers, and educators working in data science and machine learning by providing a flexible and easily accessible computing environment at no cost. Google Colab also offers the ability to run Jupyter Notebook (an open-source web app for a combination of code, formatted text, and data visualization) directly from a web browser without any configuration [15].

H. Android Studio

Android Studio is a place to create a code editor or create new code that contains features or settings to develop the usability of Android and create commands. There are several devices that must be installed in the Android Studio application, including the Android SDK and Java JDK[16].

I. Bounding box

A bounding box is an imaginary box that surrounds an identified object. The bounding box itself is in the form of a box whose size is the same as the size of the identified object. To create a bounding box itself, use the object's upper-left (UL), upper-right (UR), lower-left (LL), and lower-right (LR) pixel coordinates [17].

J. Confusion Matrix

A Confusion Matrix is a representation of the performance classification of a model in binary format that represents the performance value of the model created [18] [19].

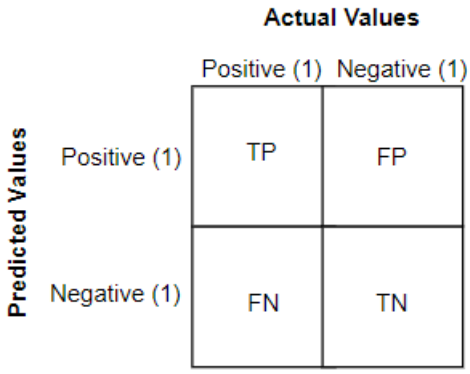


Figure 3. *Confusion Matrix*

K. Preprocessing Techniques

Preprocessing techniques are the process of preparing and transforming raw data into a more structured and analysis-ready format. This process is very important in data analysis and machine learning because it helps in producing quality and reliable data for further analysis purposes [20].

L. Labelling

Labelling is the process of providing information on image datasets by providing bounding boxes or boundary boxes. Labelling is done to obtain the coordinates of the ground-truth bounding box which will be compared with the predicted bounding box so that the Intersection over Union (IoU) value is obtained [21].

M.Roboflow

Roboflow is a web platform that has functions related to dataset collections. Roboflow is a computer vision developer framework for better data collection to

preprocessing, and model training techniques. By using Roboflow, you can share datasets while processing the dataset by annotating or marking objects to be detected using bounding boxes, in addition, pre-processing can also be used on datasets, for example doing grayscale, and augmentation using Roboflow [22].

III. RESEARCH METHOD

Data collection was done by taking pictures of healthy and cracked insulators. The images were then classified into two classes, namely the good insulator class and the damaged insulator class. Data was collected using a Canon camera, to obtain clear insulator images. After the data was collected, the data needed to be processed to remove noise and improve image quality. This process can be done using image processing software, where the process includes noise cleaning and image quality improvement.

The creation of a model to detect insulator cracks was created using object detection technology. To create the modelling software used is Google Collab. Google Collab is a cloud-based platform that allows you to write and run Python code in a browser with access to powerful computing resources, such as Graphic Processing Units (GPUs). In Google Collab, machine learning models are developed. This section includes the creation and training of models based on the data that has been processed. In the model block, there is a sub-block called Technology. Certain technologies are used to train the model. This technology is a method or procedure used to solve machine learning problems. The technologies used are MobileNetV2 and TensorFlow. MobileNetV2 is a neural network architecture optimized for mobile and embedded devices. To create efficient and lightweight models, suitable for applications on devices with limited resources. TensorFlow is an open-source framework developed by Google Brain to build and train machine learning and deep learning models. To build and train machine learning and deep learning models of artificial neural networks, the trained model is evaluated using test data. Test data is data that is not used to train the model. Model evaluation is carried out to determine the accuracy of the model in detecting insulator cracks.

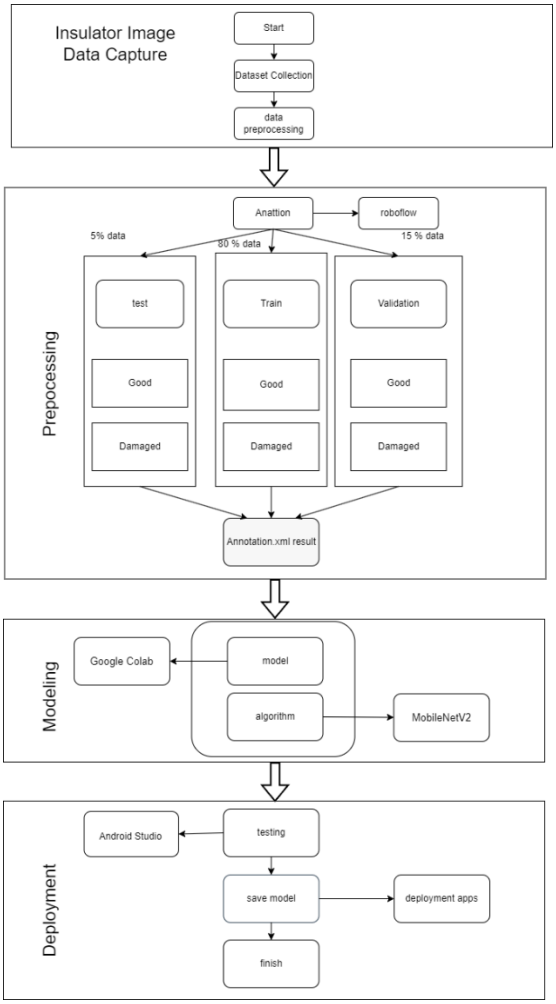


Figure 4. Procedure Chart Study

IV. RESULTS AND DISCUSSION

This study employs primary data in the form of images of medium-voltage ceramic insulators from the ULP Daya Waste

Material Warehouse. The image data was taken using a Canon EOS 700D camera and then collected into one dataset.

The dataset of ceramic insulator images that have been collected constitutes 227 ceramic insulator images with a 1:1 image capture ratio. These images include both images of insulators that are in good condition and images of insulators that are in damaged conditions.

Figure 5. Good condition insulator dataset

Figure 6. Insulator dataset in Cracked condition

A. Training Dataset Results

Dataset training results are shown in figure 7 to 10. F1-Confidence graph illustrates the superior performance of the good category over crack in terms of F1 score across varying confidence levels, highlighting its efficacy in classification tasks. Moreover, the precision-recall curve in figure 8 evaluates the classification model's performance across different threshold settings, with distinct curves representing varying rank values and their respective areas under the curve (AUC) metrics.

Finally, The Precision-Confidence Curve graph as shown in Figure 9 demonstrates the classification model's precision across varying confidence levels for the 'Bulk' and 'Rusk' classes, with an overall loss annotation at 0.918. whereas The Recall-Confidence graph compares the recall performance of two datasets, good and crack across different confidence levels, providing insights into the model's detection capabilities under varying conditions.

1. *F1-Confidence*

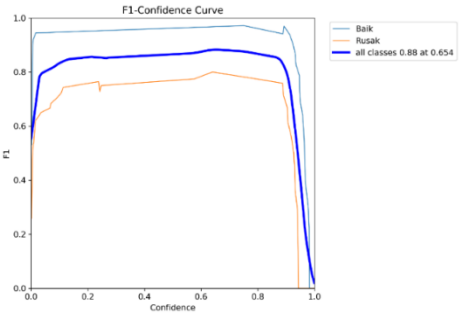


Figure 7. F1-Confidence

Figure 7 shows the results of training the dataset on Google Colab, where the following results were obtained:

1) Good Category Class (Good Class)

The blue line on the F1 Confidence curve shows that at a Confidence value of 0.1 – 0.2, the F1 score almost reaches 1, which means the model is very confident and accurate in predicting the class well. When the confidence increases from 0.2 to 0.9, the F1

score remains high and stable, indicating that the performance of the F1 score is consistent. A drastic decrease occurs once confidence exceeds 0.9, indicating that predictions with very high confidence are wrong more often. The stability model performs well at most confidence values, but a decrease above 0.9 indicates overconfidence leading to detection errors. Thresholds leave an opportunity to review and adjust confidence to improve prediction accuracy. Threshold is the limit value used to determine whether a prediction is considered a correct detection or not.

2) Damaged Category Class (crack class)

Orange lines indicate damaged scales. The F1 Curve score, which initially has a value between 0.2-0.3 (low), will increase gradually with increasing self-confidence, until it reaches peak self-confidence at a value of 0.4 to 0.6. The F1 score passes the value of 0.6, the curve fluctuates slightly and finally declines sharply at a confidence value above 0.9. A model whose confidence value is unstable is less effective in predicting damaged classes if the confidence value is low, between 0.2-0.3 (low).

3) All Classes

This thick blue line indicates the average F1 score across all classes at various levels of confidence. The highest point was in the F1 score of 0.88 at 0.654 confidence.

2. *Precision-Recall*

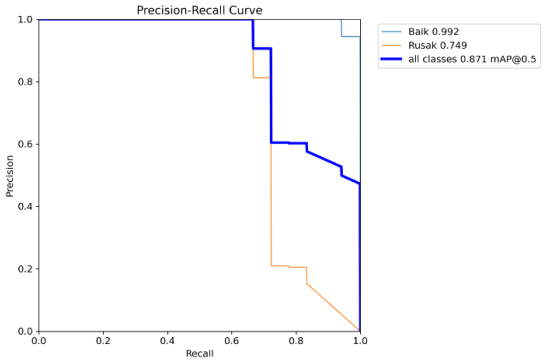


Figure 8. *Precision-Recall*

Figure 8 shows the results of training the dataset on Google Colab, where the following results were obtained:

1) Good Category Class (Good Class)

The blue line on the precision-recall curve shows that for the good class it is very high over most of the recall range, the value is almost close to 1, which means the model is very good at classifying good instances with few false positives. High precision and recall values indicate excellent performance in detecting good classes, with mAP (mean Average Precision) of 0.992. Precision decreases as recall increases closer to 1, indicating a trade-off when trying to capture all positive instances.

2) Crack Class

The orange line shows lower precision compared to fairly high recall, decreasing drastically after a certain point. This means that as the model attempts to identify all faulty instances, false positives increase significantly, reducing precision. The lower precision score indicates that the model more often misclassifies non-faulty instances as faulty, with an mAP of 0.749.

3) All Classes

The thick blue line reflects the average precision-recall across all classes, giving an overall mAP of 0.871 at a threshold of 0.5. These curves show balanced performance across all classes, but also suggest that there is room for improvement, especially in addressing classes with lower mAPs.

3. *Precision-Confidence*

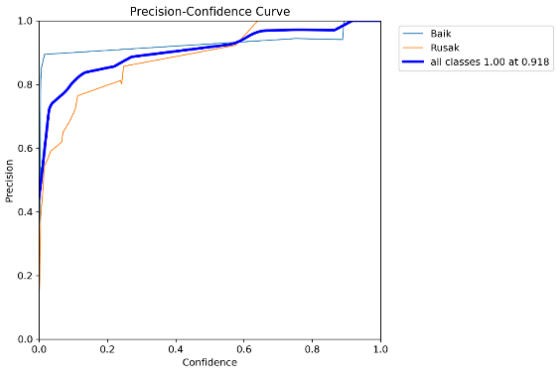


Figure 9. *Precision-Confidence*

Figure 9 shows the results of training the dataset on Google Colab, where the following results were obtained:

1) Good Category Class (Good Class)

The blue line shows the relationship between confidence and precision for the good class. Precision starts high at low confidence at a value of 0.2, and remains high and stable near 1 when confidence increases to 0.9. High precision at all confidence levels shows that the model is very reliable in identifying good classes with few false positives. Precision Stability remains high, indicating that positive predictions for the good class are very accurate even at low confidence.

2) Crack Class

The orange line shows the relationship between confidence and precision for the precision damaged class starting from a lower value at low confidence, and increasing as confidence increases to around 0.8. increasing precision with confidence models more often make prediction errors at low confidence, but accuracy increases significantly with confidence. The importance of high confidence is closely related to more accurate predictions for damaged classes

3) All Classes

The thick blue line depicts the average precision across all classes at various confidence levels. The highest precision value is 1.00 at 0.918 confidence, as shown in the legend. Optimal point At confidence 0.918, precision reaches a perfect value for all classes. Performance balance indicates that the model has optimal and reliable performance when the confidence in the predictions reaches this value.

4. *Recall-Confidence*

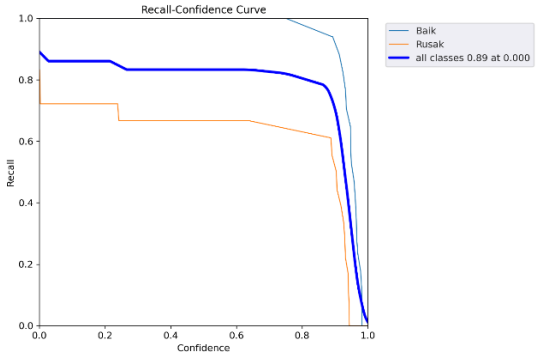


Figure 10. Recall-Confidence

Figure 10 shows the results of the training dataset on Google Colab where the following results were obtained:

1) Good Category Class (Good Class)

At low confidence up to around 0.8, recall remains high at close to 0.9, indicating that the model can detect most of the good classes

effectively. Once confidence exceeds 0.8, recall begins to decline sharply until it approaches 0. High recall at low confidence indicates that the model is able to capture many good models, but with some errors (false positives).

The drop in high confidence model may miss some good instances when only considering predictions with very high confidence, indicating a trade-off between confidence and recall.

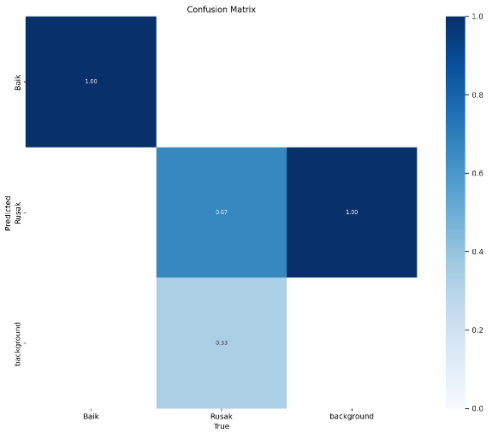
2) Crack Class

Recall is lower than the good class and is relatively constant at a level of around 0.6 to confidence approaching 0.8. There is a drastic decrease in recall when confidence exceeds 0.8. Low recall at all confidence levels indicates that the model has difficulty detecting all faulty instances, perhaps because these classes are more difficult to distinguish or because of their lack of representation in the training data. A similar trade-off of decreasing recall at high confidence suggests the model is more careful in predicting defective classes, perhaps ignoring some positive instances to avoid false positives.

3) All Classes

The thick blue line depicts the average recall across all classes, with the highest value being 0.89 at 0.000 confidence. The optimal confidence level at this point indicates the maximum recall that can be achieved when considering all classes at low confidence. The trade-off balance provides insight into how the overall model balances recall and confidence.

5. *Confusion Matrix COCO*



Gambar 11. *Confusion Matrix COCO*

In Figure 11 is the Confusion Matrix obtained from the results of training data and model creation on Google Colab. This confusion matrix consists of three classes good, damaged, and background. Following is a breakdown of the elements in this matrix.

Table 1. COCO confusion matrix results

Actual* \ *Predicted	**Good**	**Damaged**	***Background***
Good	1.0	0.33	0.00
Damaged	0.20	0.67	0.00
Background	0.00	0.00	1.0

The following is an explanation of Table 1 below:

1) True Positives (TP) The main diagonal of the matrix, shows the number of correct predictions for each class.

2) False Positives (FP) Values in columns that do not correspond to rows indicate incorrect predictions that the model classifies as classes that they should not be.

3) False Negatives (FN) Values in rows that do not correspond to columns indicate failed predictions where the correct class was not correctly identified by the model.
4) True Negatives (TN) Not directly shown in this confusion matrix, but can be calculated based on other contexts.
1. Good Category Class:
a) True Positives (TP): 1.0
The model identifies all truly good instances as a good dataset.
b) False Positives (FP): 0.33 for the Faulty class
There are a number of instances that are actually good but are predicted as damaged, indicating that the model has an error in distinguishing between good and damaged.
c) False Negatives (FN): 0.20 for Faulty class
Some defective instances were incorrectly predicted as good, which could be due to the similarity of feature characteristics between these two classes.
2. Broken Category Class:
a) True Positives (TP): 0.67
The model successfully identified 67% of the truly faulty instances.
b) False Positives (FP): 0.20 for Good class
There were instances that were incorrectly predicted as good, indicating that the model tends to perceive broken as good more often.
c) False Negatives (FN): 0.33 for good class
Indicates that the model predictions are wrong for some instances either as broken.
3. Background Class:
a) True Positives (TP): 1.0
All background instances were predicted correctly, indicating that the model has excellent ability in detecting the background.
b) False Positives (FP) and False Negatives (FN): 0.00
There is no prediction error for this class, indicating perfect prediction.

6. Model Performance Matrix Results

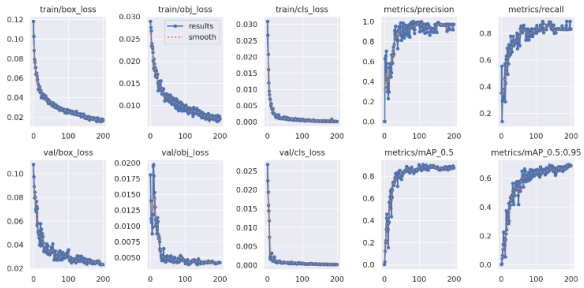


Figure 12. Model Performance Matrix

Figure 12 above displays a graph showing the performance matrix of the machine learning model during the training and validation process. Here is a detailed explanation for each chart:

1) train/box_loss:

a) This graph shows the loss in the bounding box component during training.

b) The x-axis shows the training epochs or iterations, while the y-axis shows the loss value.

c) The loss value appears to decrease significantly from around 0.12 to lower values as the epoch increases, indicating that the model becomes better at fitting the bounding box to the object.

2) train/obj_loss:

a) This graph shows the loss in object components during training.

b) The x-axis shows the training epochs or iterations, and the y-axis shows the loss value.

c) This loss value also decreases with time, indicating improvements in object detection.

3) train/cls_loss:

a) This graph shows the loss in the classification component during training.

b) The x-axis shows the training epochs or iterations, and the y-axis shows the loss value.

c) This loss value decreases from about 0.03 to lower values, indicating an improvement in object classification accuracy.

4) metrics/precision:

a) This graph shows the precision matrix during training.

b) The x-axis shows the epoch, while the y-axis shows the precision value.

c) The precision value increases as the epoch increases, indicating that the model is getting more precise in detecting objects without too many false positive predictions.

5) metrics/recall:

a) This graph shows the recall matrix during training.

b) The x-axis shows the epoch, while the y-axis shows the recall value.

c) The recall value increases as the epoch increases, indicating that the model is getting better at detecting most of the objects present.

6) val/box_loss:

a) This graph shows the loss in the bounding box component during validation.

b) The x-axis shows the training epochs or iterations, and the y-axis shows the loss value.

c) This loss value also decreases, although it fluctuates slightly more than the loss on the training data.

7) val/obj_loss:

a) This graph shows the loss in object components during validation.

b) The x-axis shows the training epochs or iterations, and the y-axis shows the loss value.

c) Loss values decrease over time, indicating improved model performance on validation data.

8) val/cls_loss:

a) This graph shows the loss in the classification component during validation.

b) The x-axis shows the training epochs or iterations, and the y-axis shows the loss value.

c) The loss value decreases, indicating an increase in classification accuracy on validation data.

9) metrics/mAP_0.5:

a) This graph shows the mean Average Precision (mAP) at a threshold of 0.5 during training.

b) The x-axis shows the epoch, while the y-axis shows the mAP value.

c) The mAP value increases as the epoch increases, indicating that the model becomes more accurate overall in detecting and classifying objects.

10) metrics/mAP_0.5:0.95:

a) This graph shows mAP at various thresholds from 0.5 to 0.95 during training.

b) The x-axis shows the epoch, while the y-axis shows the mAP value.

c) This mAP value also increases with increasing epochs, indicating an increase in overall model accuracy at various detection thresholds.

B. Detection Results Use Application

1. Good insulator

Figure 13. Below shows that the application is scanning or assessing the condition of the isolator. The results show that the isolator is still in good condition with a reading or confidence level of 98.04%. The screen is surrounded by a blue line indicating the detection area of the object being analysed by the application.

Figure 13detection area by Android Phone.

2. Damaged insulator

Figure 14 below is a reading of the application that is analysing the condition of an electrical insulator. In this image, there are two areas marked with red and blue boxes. The red box marks an area that has a reading of damaged conditions with a confidence level of 73.56%. The blue box indicates another area that also has a reading of damaged conditions with a higher confidence level, namely 99.44%. This marker indicates that the application has detected damage to the insulator at several different points. Each area of damage has a different level of confidence, indicating how confident the application is in identifying the damage.

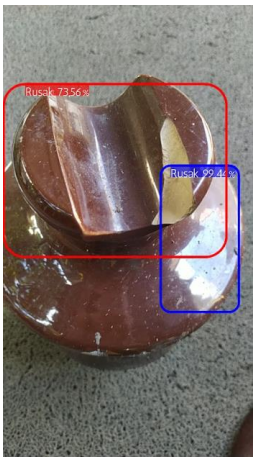


Figure 14. The cracked of insulator detection result

3. Insulator detection results in the medium voltage network

Figure 59 below is the application detection results on insulators installed in the medium voltage network where the detection results obtained are that the insulator is damaged with a reading accuracy value of 73.37%.

Figure 15. Detection results on medium voltage networks

V. CONCLUSION

Based on the results of the discussion that has been done, the conclusions of this study are:

1. The medium voltage insulator crack detection model was successfully designed and implemented using MobileNetV2 and TensorFlow technology.
2. The accuracy of the model in the COCO matrix shows that the good class category has a reading value of 1.0, the damaged class is 0.67 and the background is 1.0, and the performance graph shows that the model can read objects well because the reading curve is in line with the smooth curve.
3. MobileNetV2 and TensorFlow technology as an insulator crack detector has been successfully applied to an Android mobile phone.

REFERENCE

1. Thanh, P.N., M.-Y. Cho, and T.N. Da, *Insulator leakage current prediction using surface spark discharge data and particle swarm optimization based neural network.* Electric Power Systems Research, 2021. 191: p. 106888.
2. Xing, Y., et al., *Defects and failure types of solid insulation in gas-insulated switchgear: In situ study and case analysis.* High Voltage, 2022. **7**(1): p. 158-164.
3. Rouhanizadeh, B. and S. Kermanshachi, *Development of strategies to prevent third rail insulator failures in transit systems.* Urban Rail Transit, 2021. **7**: p. 58-70.
4. Jadid, R., *Thermal Imaging on Ceramic and Glass Insulators against Aging* . 2023, Gadjah Mada University.
5. Dewi, T.p.,*Analysis of Insulation Resistance Reduction of Porcelain Pin Type Insulation of Sungkai Feeder of Bungaran Main Substation at Ampera Ulp Pt Pln (Persero)* . 2021, Sriwijaya State Polytechnic.
6. Abdul Malik, A., *Human Face Identification Using Yolo Frameworks With Scale Modifier Method As Real Time Preprocessing.* 2023.
7. Al Choedori, M.M., *Penerapan Image Processing Untuk Mendeteksi Pergerakan Ikan Koki Dengan Metode Cascade Classifier.* 2021.
8. Alfarizi, D.N., et al., *The Use of YOLO Method in Object Detection: A Systematic Literature Review.* AI and DSS: Journal of Artificial Intelligence and Decision Support Systems, 2023. **1** (1): p. 54-63.
9. Marine, Y., *Implementation of Canny Edge Detection Algorithm Using Python and Opencv.* Smart Techno (Smart Technology, Informatics and Technopreneurship), 2023. **5** (1): P. 1-7.
10. Barokah Asmarahman, T., *Detection and Identification of Traffic Signs Based on You Only Look Once Algorithm* . 2023, Muhammadiyah University of Jakarta.
11. Li, Y., et al., *YOLO-ACN: Focusing on small target and occluded object detection.* IEEE access, 2020. **8**: p. 227288-227303.
12. ANHAR, A. and R.A. PUTRA, *Design and Implementation of Self-Checkout System in Retail Stores using Convolutional Neural Network (CNN).* ELKOMIKA: Journal of Electrical EnergyEngineering, Telecommunication

Engineering, & Electronic Engineering, 2023. 11(2): p . 466.
13. Darwis, M.R., *Facial Expression Recognition on Video Based on Deep Learning Using Mobilenet V2* . 2022, Sultan Syarif Kasim State Islamic University, Riau.
14. Ae, N.H. and M.I. Zul, *Indonesian Sign Language Translator Application to Voice Based on Android Using Tensorflow*. Journal of Applied Computer, 2021. **7** (1): p. 74-83.
15. Febrywinata, E., *Introduction and Classification of Fruit Types Using the CNN Method in a Simple Way Using Google Colab*. Mercury: Journal of Information Systems and Informatics Engineering Research, 2024. **2** (4): p. 185-193.
16. Holide, YA, A. Krismon, and DZE Prasetya. *Android-Based Kediri City Tourism Application Using Android Studio* . in *National Seminar on Technology & Science* . 2022.
17. Putra, B., B. Nugroho, and F. Anggraeny, *Use of elevators in buildings-Detection and counting of humans using Yolo-CNN*. Journal of Informatics and Information Systems, 2021. **2** (1): p. 67-76.
18. Sharma, L. and M. Carpenter, *Computer Vision and Internet of Things: Technologies and Applications*. 2022: CRC Press.
19. Chakraborty, R., A. Ghosh, and J.K. Mandal, *Machine Learning Techniques and Analytics for Cloud Security*. 2021: John Wiley & Sons.
20. Kamila, NL, *Understanding What Data Preprocessing Stages Are.* 2023.
21. Sugandi, AN and B. Hartono. *Implementation of Image Processing on Quadcopter for Human Detection Using YOLO Algorithm* . in *Proceeding Industrial Research Workshop and National Seminar* . 2022.
22. Hayati, NJ, D. Singasatia, and MR Muttaqin, *Object Tracking Using the You Only Look Once (Yolo) V8 Algorithm to Count Vehicles.* Komputa: Scientific Journal of Computer and Informatics, 2023. **12** (2): p. 91-99.