Monitoring of PV Modules and Hotspot Detection Using Convolution Neural Network Based Approach

B. Sandeep, D. Saiteja Reddy, R. Aswin, R. Mahalakshmi

Monitoring of PV Modules and Hotspot Detection using Convolution Neural Network based Approach

[1]B Sandeep, [2]D Saiteja Reddy, Aswin R, *R Mahalakshmi

Department of Electrical and Electronics Engineering,
Amrita School of Engineering, Bengaluru,
Amrita Vishwa Vidyapeetham,
India.
[1]sandeep2001b@gmail.com, [2]tejadevireddy123@gmail.com, *d_mahalakshmi@blr.amrita.edu

Abstract: The use of solar photovoltaic systems in green energy harvesting has increased greatly in the last few years. Fossil fuels reaching the end are also growing rapidly at the same rate. Despite the fact that solar energy is renewable and more efficient, it still needs regular Inspection and maintenance for maximizing solar modules' lifetime, reducing energy leakage, and protecting the environment. Our research proposes the use of infrared radiation (IR) cameras and convolution neural networks as an efficient way for detecting and categorizing anomaly solar modules. The IR cameras were able to detect the temperature distribution on the solar modules remotely, and the convolution neural networks correctly predicted the anomaly modules and classified the anomaly types based on those predictions. A convolution neural network, based on a VGG-based neural network approach, was proposed in this study to accurately predict and classify anomalous solar modules from IR images. The proposed approach was trained using IR images of solar modules with 5000 images of generated solar panel images. The experimental results indicated that the proposed model can correctly predict an anomaly module by 99% on average. Since it can be costly and time-consuming to collect real images containing hotspots, the model is trained with generated images rather than real images. A generated image can be used more efficiently and can also have custom features added to it. In the prediction process, the real image is processed and it is sent to the model to determine bounding boxes. It provides a more accurate prediction than direct use of the real image. Here we have used CNN custom model and TensorFlow libraries.

Keywords: Convolution neural network, PV modules, Detection of hotspots, CNN, TensorFlow, Solar Panels.

1 Introduction

The benefits of solar photovoltaic systems are numerous, and they are extremely useful. Their uniqueness lies in their lack of moving parts (in the sense of classical mechanics). There is no leakage of fluids or gasses (except in hybrid systems). They do not require fuel, which is one of the best features of these systems. They respond rapidly and achieve full output immediately [1]. Although the cells run at moderate temperatures and do not emit pollution while producing electricity. Solar cells require little

maintenance if properly manufactured and installed. Solar energy has gained a reputation as a clean, renewable energy source. Solar energy can help to reduce carbon dioxide levels in the atmosphere. Solar energy has no negative impact on the environment [2]. Solar power requires no additional resources than a source of light to function, in addition to not emitting greenhouse gasses. As a result, it is both safe and environmentally friendly.

PV cells and solar energy, like other renewable energy sources, have intermittent difficulties. It cannot be transformed into power constantly at night or in foggy or rainy conditions [3]. In other words, PV cells may not be able to generate enough electricity to fulfill the peak demand of a power grid. Solar energy panels are a less reliable power source because of their intermittency and instability. In addition to PV cells, inverters and storage batteries are required. An inverter is a device that turns direct electricity into varying electricity that may be used on your power grid. Storage batteries help on-grid connections because they provide a consistent supply of electric power. On the other side, this greater expenditure may be able to address the PV cells' intermittency issues [4]. PV plants' ability to produce energy may be affected by the failure of modules or components (wiring, bypass diodes, etc.), malfunctions, or accumulated dust and soiling on their surfaces; the last condition can decrease efficiency by more than 18% compared to normal conditions. For example, consider the PV system installed in Pavagada solar park (Karnataka) with a size of 53 square kilometers (13000 acres). When working at its best, it can produce almost 2350 MWh per year. Assuming that the system operates normally with soiled modules due to dust deposits, debris, and bird droppings, the energy production could be decreased by 300 MWh/year, or 14.64% [5].

For PV systems to perform optimally, monitoring systems should be used to detect anomalies in normal operation and to check the cleanliness of modules using periodic cleaning. Using a convolution neural network to monitor solar modules with an infrared camera is performed. In addition to being able to monitor the condition of photovoltaic modules and other components, this technology can be used for pre-diagnosis and diagnosis of faulty working conditions by using UAVs (drones), electric equipment in PV systems [6],[7]. CNN can be used to detect any flaws or malfunctions in PV modules, e.g., conditions that result in local overheating, without interfering with the usual operation of the system [8]. Dirt, dust, and bird droppings on the surface can cause local overheating by changing the thermal exchange conditions between the module and the environment. In this work, machine learning is applied to construct a tool that automatically identifies abnormal operating conditions of PV modules [9]. The paper proposes to use machine learning as a tool in this work. In this project, we will develop a system for automatic classification of thermograms by using a convolutional neural network (CNN), distinguishing between overheating caused by external factors or by

faults on the surface of the photovoltaic module [10]. Thermographic images can now show an anomaly state caused by a fault [11]. The importance of photovoltaic energy generation, the benefits of this energy generation, the various methods for identifying power losses in this generation method, and, most importantly, the most commonly raised problem of PV hotspots generation in PV modules and the causes of these hotspots are all discussed in the above literature surveys [12],[13].

Therefore, the solar panel must be monitored for failures, particularly hotspots. This paper proposes a method to identify hotspots using TensorFlow. Hotspot images are generated using the same processing method as solar panel images. Training models are validated with validating images, followed by testing of images in which the model determines the hotspot type and localization of the hotspot. In real solar panel images, the model can detect and classify the type of hotspot and its location based on the training with different types of hotspots containing images and subsequent validation for any particular generated image. In order to prevent panel damage, action needs to be taken immediately based on an analysis of the hotspot type and location.

2 SYSTEM MODEL

2.1 Introduction

The input data is generated in python and after that, it is split into train data and test data. Now the model has sufficient data and it is split into train data and test data so that better accuracy can be obtained. In below figure 4.1, the process of the project is shown.

2.2 Block Diagram

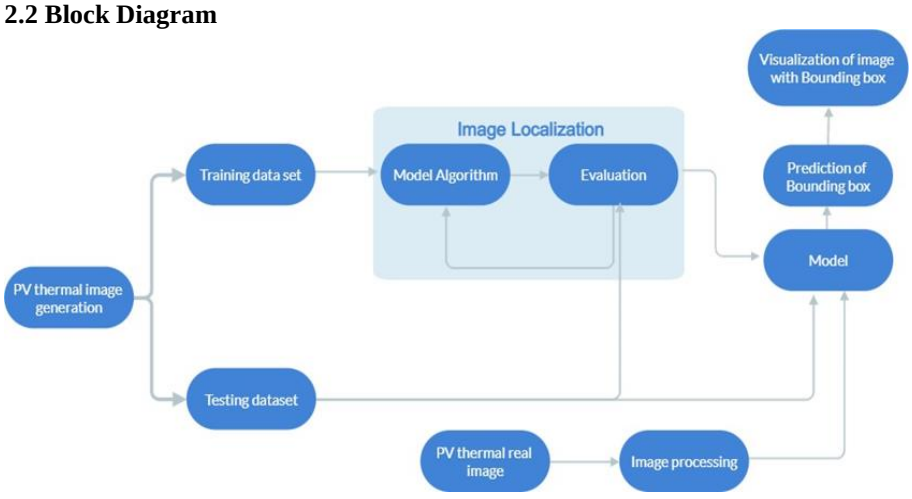

Fig. 1. Block Diagram for Hot Spot localization.

Fig. 1 shows a block diagram of the hotspot localization process, which includes all the key steps. It starts with the data generation, then it splits into training and testing data sets. Then comes the algorithm, and then after that, the evaluation, which is nothing but checking the model performance. The model is again trained and improved, and this is how Image Localization is achieved. Following that, real images are converted into processed images and fed into the model. Next, the Image localization model predicts the bounding box is followed by analysis and visualization of the results.

2.3 Data generation

PV images with hot spots are generated through Python with an image ratio of 120 X 240 pixels. During generation, 10,000 unique images were generated and split into 5000 training images and 5000 test images. Each image size is 120 X 240 pixels.

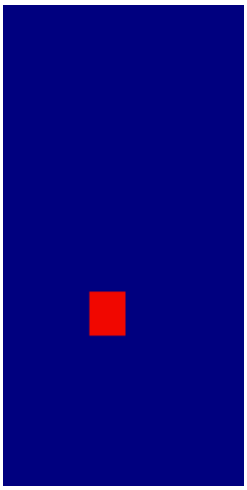

Fig. 2. Generated Image.

```
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
    <folder>images</folder>
    <filename>images/training/images1.png</filename>
    <path>C:images/training/images1.png</path>
    <size>
        <width>120</width>
        <height>240</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>spot</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>43</xmin>
            <ymin>142</ymin>
            <xmax>68</xmax>
            <ymax>167</ymax>
        </bndbox>
    </object>
</annotation>
```

Fig. 3. XML file.

PV thermal image is shown in Fig 2 along with the XML file for the bounding box label in Fig 3. With a dark blue background and a red hotspot at a random location, the generated image is similar to a real image. The bounding box is a box containing the hotspot location in the given image. XML files are used to view the image with the bounding box.

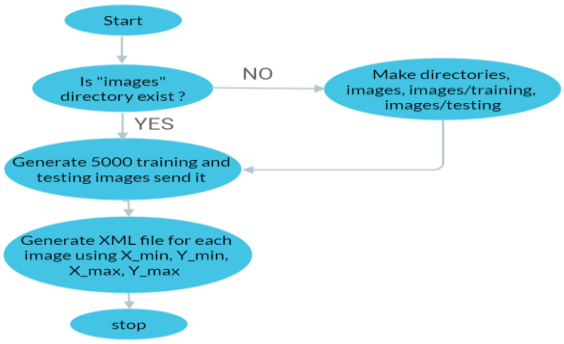

Fig. 4. Data Generation flowchart.

Fig. 4 shows a flowchart for Data Generation. if there is no images directory, it creates directories such as images, images/training, images/testing to hold the images in the relevant directory. Python was used to create the generated images. 10,000 unique images were generated during the generation process and then divided into 5000 training and 5000 test datasets. These are converted into Tensorflow datasets. Once we have converted these into Tensorflow datasets, the model takes the test and training datasets separately. In addition to images, XML file coordinates are also generated, including X_min, Y_min, X_max, and Y_max. These coordinates can be used to draw bounding boxes. These XML files are also placed in directories along with images. For training and validation of models, these XML files serve as labels for images.

2.4 Training and Testing Data

In the second stage, when data is generated, the model must be trained and tested to improve accuracy and results. The generated data is separated into two subgroups, as seen below. During generation, 10,000 unique images were generated and split into 5000 training images and 5000 test images. With 5000 training images, we were able to improve the model's accuracy. Test images could be any number for validation purposes only. We chose the 5000 testing images. The generated images were created programmatically using Python.

2.4.1 Training Dataset

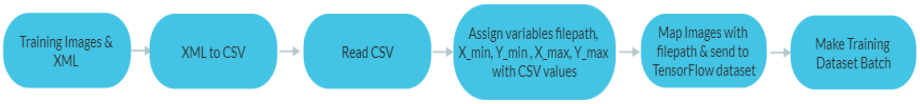

Fig. 5. Training Dataset.

Fig. 5 shows a flowchart for Training Dataset. The training images with XML files are converted into CSV as they need to be understandable by TensorFlow then the assigning of bounding box coordinates is done in CSV values and after that, Mapping of images with the file path and sending it to TensorFlow. Now as the data set is created

now, the data is divided into batches each batch containing 64 images and fed to the model.

2.4.2 Testing Dataset

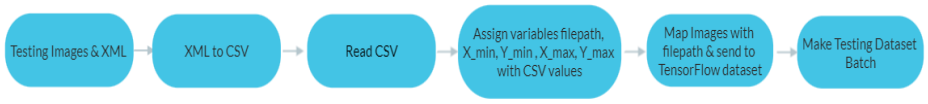

Fig. 6. Testing Dataset.

Fig 6 shows a flowchart for Testing Dataset. Similar to the training data the test data also will be there as the data set used to offer an unbiased assessment of a final model's fit to the training dataset. The test dataset serves as the gold standard against which the model is judged. It's only used once a model has been fully trained (using the train and validation sets). The test set is typically used to compare models (For example, for many Kaggle competitions, validation and training sets are released initially together, then the test set is released toward the end of the competition and the model's performance on the test set determines the winner). The validation set is frequently used as the test set, but this is not a good practice. The test set is well-curated in general. It includes properly sampled data from a variety of classes that the model might encounter in the actual world.

2.4 Image Processing

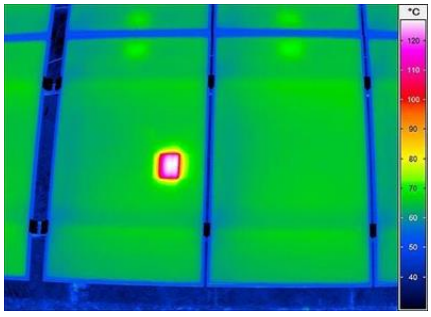

Fig. 7. Real PV Thermal Image (Source: Internet)

As the model has to detect hotspots in real PV thermal images like the one in Fig 7, the model has been trained using the generated data and validated. It is necessary to process the real images (in a similar way to the generated image) so that the model can understand them. Hence, the real PV thermal images must be processed.

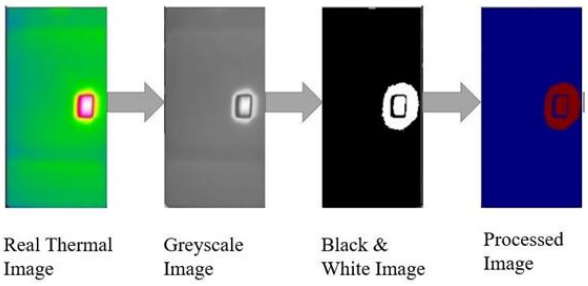

Fig. 8. Real thermal image to Processed image.

Fig 8 shows how real images are converted into processed images by Image Processing.

- Real Thermal Image
- Grayscale Image
- Black & White Image
- Processed Image

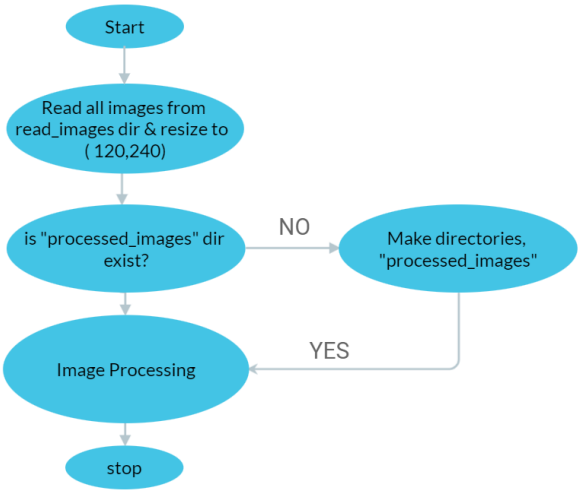

Fig. 9. Processed Image Generation.

Fig. 9 shows the processed image Generation. First, all real images in the real_images directory are read and resized to 240 x 120. The processed_images directory is created if it does not exist and every image is sent to the image processing function, after which the processed images are sent to the processed_images directory.

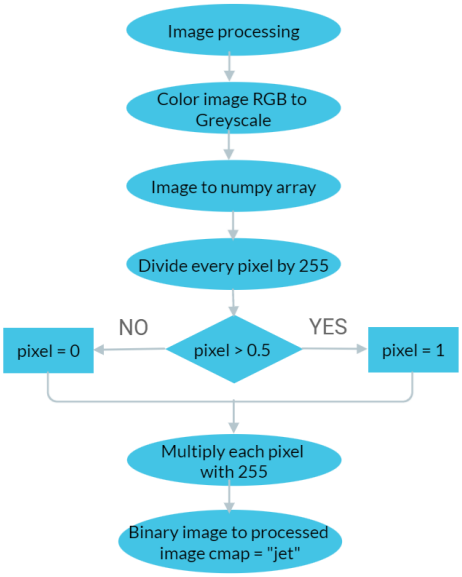

Fig. 10. Image Processing.

Fig. 10 shows how real color images are converted into grayscale images by using a filter. The image is converted into a NumPy array and then each pixel is divided by 255, and then the result is compared with 0.5. If the pixel is greater than 0.5, the set pixel value equals 1 then it is black, and else the pixel is 0 then it is white. 255 is multiplied by each pixel again. This logic transforms the grayscale image into a Black & White image (Binary image). A binary image is converted to a processed image by using c-map as "jet".

2.5 Image Localization

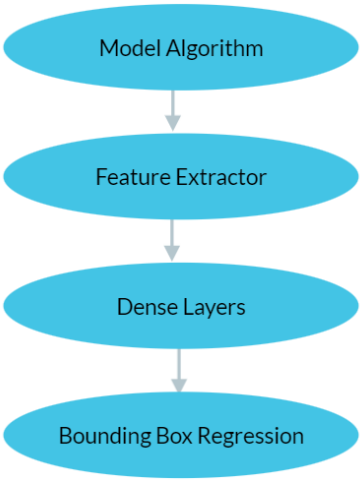

Fig. 11. Image Localization Model Algorithm.

Fig 11 illustrates how the model algorithm works, which includes the feature extractor and dense layers followed by the bounding box regression. Tensorflow Training batches are sent to a feature extractor that has Conv2d filters and average pooling filters for extracting key features that are required to train the model. In the Dense layer, which follows the feature extractor, each node takes the flattened image output and multiplies with weights, adds bias, and sums all of these to the activation function, resulting in outputs of the function. The bounding box is predicted by adjusting the weights and biases in the dense layer. A Bounding Box regression has four nodes and outputs four values at the same time: X_min, Y_min, X_max, and Y_max.

2.6 Model Flowchart

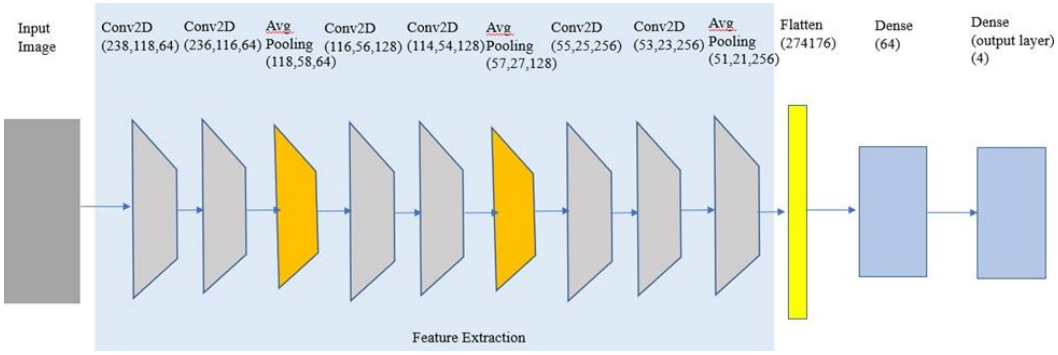

Fig. 12. Image Localization Model Flowchart.

The model flow chart in Fig. 12 shows the model layers and the resulting architecture which is inspired by the VGG architecture, which is two convolutions followed by average pooling, and the cycle is repeated three times. Initially, there are 64 filters, followed by 128, and finally, 256 filters. It shows the flow of the model algorithm.

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 240, 120, 3)]     0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 238, 118, 64)      1792
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 236, 116, 64)      36928
_________________________________________________________________
average_pooling2d_2 (Average (None, 118, 58, 64)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 116, 56, 128)      73856
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 114, 54, 128)      147584
_________________________________________________________________
average_pooling2d_3 (Average (None, 57, 27, 128)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 55, 25, 256)       295168
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 53, 23, 256)       590080
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 51, 21, 256)       590080
_________________________________________________________________
flatten_1 (Flatten)          (None, 274176)            0
...
=================================================================
Total params: 19,283,076
Trainable params: 19,283,076
Non-trainable params: 0
```

Fig. 13. Model Summary.

This Fig. 13 shows that the model summary displays all of the model's layers as well as their output shapes. It takes a 240*120-pixel input image. The size of the output image is reduced by 2x2 in each conv2d layer using the 3x3 Filter. The resulting image's size is reduced by half for each Average pooling layer. The output shape after flattening is 274176. This will be the input for the 64 neutron dense layers. The last dense layer has 4 neutrons.

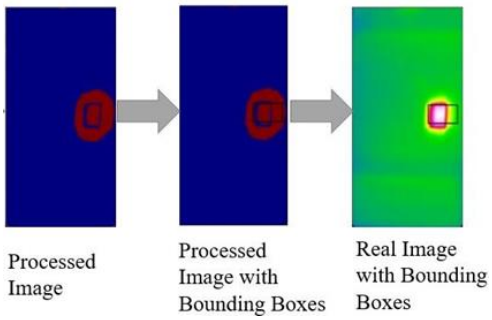

Fig. 14. Predicting bounding box for processed image and real image

Fig 14 shows when the processed image is sent to the model it predicts output X_min, Y_min, X_max, Y_max for drawing the Bounding Box. By using these values, a bounding box can be drawn for the processed image as well as the real image.

3 Results

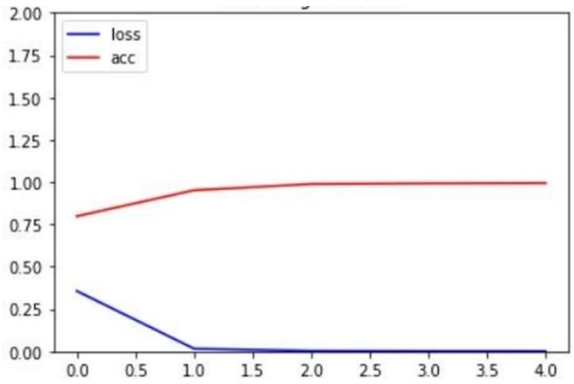

Fig. 15. Accuracy and Bounding Box Loss Results.

```
Validation accuracy:  0.9948666691780009
Validation loss:  0.00077322393190115569
```

Fig. 16. Validation Accuracy and Loss Values

The results are shown in Fig.15 as in graphs i.e. bounding box loss and accuracy. In Fig.16 the validation accuracy and loss results are given. An epoch is one training iteration, so in one iteration all samples are iterated once. Fig. 15 shows the graphs of Bounding Box Loss and accuracy. Fig 16 shows validation accuracy and loss. As it is observed the validation accuracy is 0.99486. The model is working perfectly

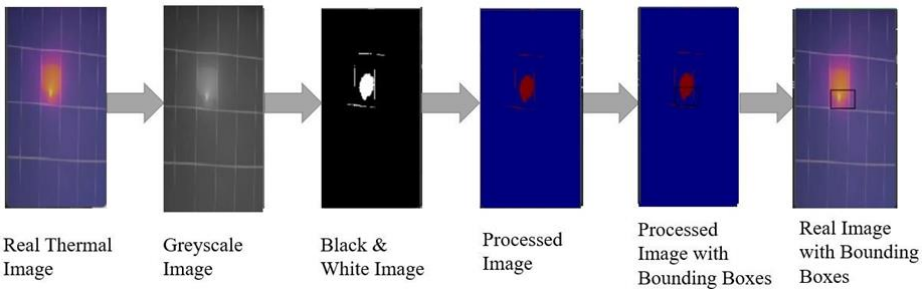

Fig. 17. Step by step process through images

The Total process of this paper is shown in Fig.17 as initially the PV thermal image is converted into a grayscale image using some filters and then based on the pixel values the grayscale image is converted into a black & white image. Now, this image is converted into a processed image using the model algorithm which is our model understandable image format. As the model is trained with this type of data set it can

easily detect the hotspot and draw bounding boxes around it. And based on the bounding box coordinates the model can draw the same boxes in the real thermal image also. Thus, the hotspot is detected in the real PV thermal image and the model detection accuracy is good at 0.994.

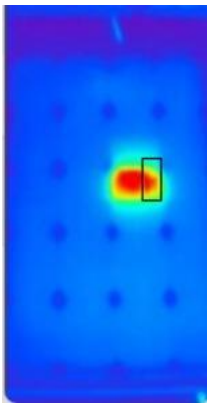

Fig.18a Real image 1

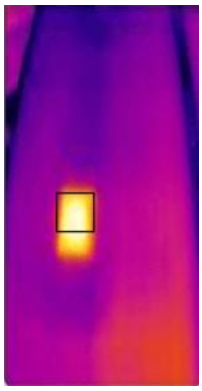

Fig.18b Real image 2

These are the bounding box results for other images.

4 Conclusion

There is an analysis of the accuracy results of both the training and testing data sets, and for the hotspot, localization is done by drawing bounding boxes, and the real PV image with hotspots is converted to the generated image format, and finally, according to the bounding box coordinates, the hotspots are detected in real thermal images.

With the generated images of PV modules, the model is trained. It is then used to process the real images of PV modules. Real thermal images are converted into grayscale and then into black and white. After that, the processed images are generated using a color map jet. As the model is trained with 5000 images the accuracy is very good. Thermal images can be converted to processed images, which is the format that is used to train the model, and the bounding boxes can be easily drawn around the hotspot. After getting the bounding box coordinates the same boxes can be drawn on a real image. So, the hotspot detection and localization for real thermal images is done

5 References

1. M. Gautam, S. Raviteja, R. Mahalakshmi, “Household Energy Management Model to Maximize Solar Power Utilization Using Machine Learning”, Procedia Computer science, Volume 165, pages 90-96, 2019.

2. Sophie N, Zahra Barbari “Study of defects in PV modules”, Energies 2019, pp - 107.

3. Mani, M.; Pillai, R. Impact of dust on solar photovoltaic (PV) performance: Research status, challenges, and recommendations. Renew. Sustain. Energy Rev. 2010, 14, 3124–3131.

4. Kudelas, D.; Taušová, M.; Tauš, P.; Gabániová, L’.; Košˇco, J. Investigation of Operating Parameters and Degradation of Photovoltaic Panels in a Photovoltaic Power Plant. Energies 2019, 12, 3631.

5. Cristaldi, L.; Faifer, M.; Rossi, M.; Catelani, M.; Ciani, L.; Dovere, E.; Jerace, S. Economical Evaluation of PV System Losses Due to the Dust and Pollution. In Proceedings of the 2012 IEEE International Instrumentation

6. Alsafasfeh, M.; Abdel-Qader, I.; Bazuin, B.; Alsafasfeh, Q.H.; Su, W. Unsupervised Fault Detection and Analysis for Large Photovoltaic Systems Using Drones and Machine Vision. Energies 2018, 11, 2252.

7. X. Li, Q. Yang, Z. Lou, and W. Yan, ‘‘Deep learning-based module defect analysis for large-scale photovoltaic farms,’’ IEEE Trans. Energy Convers., vol. 34, no. 1, pp. 520–529, Mar. 2019, doi: 10.1109/tec.2018.2873358.

8. Xudie Ren, Haonan Guo, Shenghong Li, Shilin Wang and Jianhua Li. A Novel Image Classification Method with CNN-XGBoost Model.’’, C. Kraetzer et al. (Eds.): IWDW 2017, LNCS 10431, pp. 378–390, 2017.

9. Zhao, Q.; Shao, S.; Lu, L.; Liu, X.; Zhu, H. “A New PV Array Fault Diagnosis Method Using Fuzzy C-Mean Clustering and Fuzzy Membership Algorithm.” Energies 2018, 11, 238

10. Pei, T.; Hao, X. A Fault Detection Method for Photovoltaic Systems Based on Voltage and Current Observation and Evaluation. Energies 2019, 12, 1712

11. Jadin, M.S.; Taib, S. Recent progress in diagnosing the reliability of electrical equipment by using infrared thermography. Infrared Phys. Technol. 2012, 55, 236–245

12. P. G. Shivani, S. Harshit, C. V. Varma, and R. Mahalakshmi, "Detection of Broken Strands on Transmission Lines through Image Processing," 2020 4th International Conference on Electronics, Communication, and Aerospace Technology (ICECA), Coimbatore, India, 2020, pp. 1016-1020.

13. P. G. Shivani, S. Harshit, C. V. Varma, and R. Mahalakshmi, "Detection of Icing and Calculation of Sag of Transmission Line Through Computer Vision," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2020, pp. 689-694.