Detecting Dirty Solar Panels

Toby Cheng, Jack Bellamy, Quincy Sproul, Alex Bott, Adam Abuobaid

***Abstract*—Solar power is a vital source of renewable energy, and with climate change continuing to rise, it is of paramount importance that solar panels perform efficiently to limit reliance on fossil fuels. Bird droppings, dust, pollution and tree sap can all impact the performance of solar panels, reducing their reliability as a source of green energy. In this paper, we evaluate the use of artificial intelligence to determine whether a given solar panel needs cleaning, using methods such as K-Nearest Neighbors, Convolutional Neural Networks and Support Vector Machining. Each method will have its effectiveness measured using a variety of performance metrics, and its limitations and potential improvements discussed.**

An additional object detection model for use with drone footage will be explored. The model detects whether a dirty of clean solar panel is present in a frame using a fully convolutional YOLOv8 object detection system.

I. Introduction

Solar panels have been used for electricity generation since the mid-20th century but were initially expensive to construct and maintain. However, advances in technology since then has helped reduced these costs, making solar panels a viable source of energy. Increased awareness of the environmental benefits of solar energy has meant that in many parts of the world, governments encourage the installation of solar panels on homes and many people are choosing to install them as a way to reduce their carbon footprint and save money on energy bills.

In a recent study by Sulaiman [1], analysis on the effects of the accumulation of dirt, dust, water, sand and moss on solar panels was carried out. It was found that the build up of debris could decrease performance by up to 85%, with sand being the most detrimental factor.

As such, consistent maintenance must be carried out in order to maintain a solar panels performance. Large solar fields are often constructed in fields, where loose dirt is likely to cover the panels, particularly when high winds are present. To maximise energy production, we aim to create an object detection system for use with drone footage, that will determine which solar panels are clean and dirty. Using artificial intelligence will reduce the time, cost and energy required to locate dirty solar panels, when compared to manual identification.

II. Literature review

A study by TechLabs [2] used deep learning to detect the presence of soil on solar panels. A CNN model was used to identify and quantify the build up of dirt. After training, the proposed CNN achieved an accuracy of 90%, but was only able to detect small quantities of soil. Khandakar [3], looked at the use of machine learning to record temperature, humidity and solar radiation data to predict the energy yield of a given solar panel. 80% of the model predictions were within 10% of the target value. Another study by Zyout [4], explored automatic inspection of solar panels using pre-trained CNN models, achieving a best validation accuracy of 93.3%.

These studies have demonstrated the potential of using artificial intelligence to detect and monitor the cleanliness of solar panels. The use of machine learning and deep learning algorithms can help improve the performance of solar panels and reduce maintenance costs. Further research can be completed to optimize the accuracy and efficiency of these artificial systems, which could develop new, more efficient approaches for monitoring and cleaning solar panels.

III. Dataset

The solar panel dataset [5] used to train and test models in this report was found on Kaggle, chosen as it was the largest openly available dataset applicable to our project aim. Some images in this dataset were found to have watermarks or unrelated images which had to be manually removed, leaving a final collection of 1213 clean and 840 dirty solar panel images. A random image from each class is shown in Figure 1. It should be noted that this size of the dataset is relatively small, and the limitations due to this are discussed in part VI.

Fig. 1: An example image from each class in the dataset used in this report [5]. The image on the left is labelled clean, and the image on the right is labelled dirty.

IV. Method

Identifying whether a solar panel is clean or dirty is a binary image classification problem, and there are a variety of unsupervised & supervised machine learning techniques that are applicable to this task. In each classification model, a 55%-15%-30% random split from the whole data set was used to determine the training, validation and test sets respectively. Some of the key advantages and limitations of each model are mentioned in this section, but the overall performance is discussed and evaluated in section V. Additionally, a seperate object recognition model for use with drone footage was explored, using a single stage detector.

A. K-Nearest Neighbors

The simplest non-parametric classification method is K-Nearest Neighbors (KNN). A KNN algorithm will classify unknown data points by calculating the majority class among the k closest neighbors. The value of k is a hyper-parameter that can be adjusted, and will affect performance based on the given training data set.

A distance metric must therefore be determined that calculates what the nearest neighbors are. A study by Chomboon *et al* [6], found that Euclidean distance had one of the highest prediction accuracy's in KNN classifying tasks. As such, this metric was chosen for the KNN model, calculated using equation (1). The variable x represents the feature vector of the data point we aim to classify, and the variable y is the feature vector of a data point with a known label.

$$D_{Euclidean}(x, y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{1}$$

In the case of image classification, the feature vectors are formed from a reshaped matrix of the red, green and blue (RGB) values for each pixel. Thus, a separate algorithm for each RGB colour was ran, using the respective colour's reshaped feature vector. Each algorithm iterates through all training data points calculating the euclidean distance between the feature vector and the unclassified data point. The majority prediction from each colour's KNN algorithm is used as the final class prediction.

Hyper-parameter tuning

The KNN method has a single hyperparameter k that can be adjusted. The model was ran for integer values of $k = 1$ to $k = 7$, and its accuracy was recorded, shown in Figure 2.

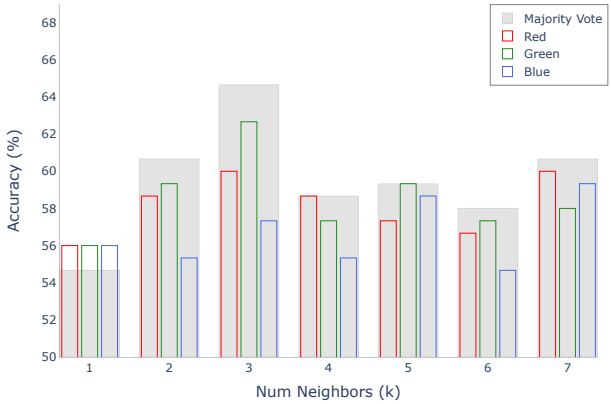

Fig. 2: A bar plot showing different values for k and the respective model's accuracy, shown by the grey bar. Additionally, the performance of each individual colour algorithm is shown. $k = 3$ is shown to be the most accurate model, correctly classifying 64.7% of data points from the test set.

As KNN is a simple algorithm that performs no actual learning, the performance of the optimized model is - as expected - relatively poor. A visualization of predictions for the green value algorithm is shown in Figure 3.

KNN suffers from the 'curse of dimensionality' [7], a term used to describe the limitations of the method in high dimensional data sets such as the one used in this report. In high dimensional spaces, it becomes highly computationally demanding, and additionally the assumption of similar data points being located close to one another breaks.

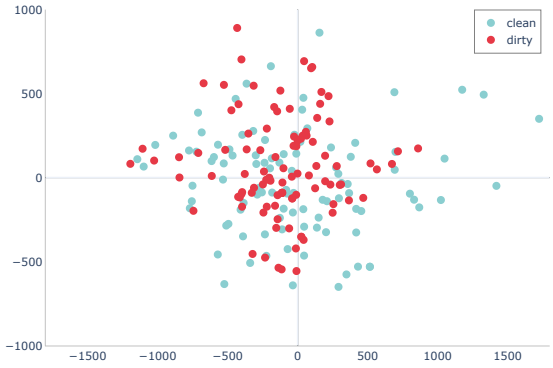

Fig. 3: Visualising a subset of 200 predictions from the $k = 3$ KNN, using principle component analysis (PCA).

B. Convolutional Neural Network

Another classification technique explored in this report is a convolutional neural network (CNN), which is a type of deep learning neural network applicable to image recognition. The proposed CNN uses 8 different types of layers in the architecture of a visual geometry group (VGG16). VGG16 is a powerful CNN that has been widely used in various computer vision tasks such as object recognition, image classification, and segmentation [8]. The first layer is the input layer which accepts the image data, followed by a convolutional layer which performs element wise multiplication to extract features using a kernel of size 3x3x3, a process repeated for every pixel of the image. This layer captures simple features like edges and blobs, while the deeper convolutional layers will capture more complex features like textures and object parts. The third layer is an activation layer, using ReLU activation. ReLU sets negative values to zero and keeps positive values unchanged, which introduces non-linearity to the network helping to learn complex patterns. The two aforementioned layers are repeated five times. The fourth layer is a pooling layer, which reduces the spatial dimensions of the features from the convolutional layer. This is followed by a flatten layer, which takes a multi-dimensional array and flattens it into a one-dimensional vector. This is to get the data in the correct input shape for the next layer. Next is a dropout layer, a regularization technique

to prevent overfitting. It achieves this by randomly setting a fraction of the inputs to zero during training, meaning some of the neurons have dropped out. A dense layer is then used which connects the output of the previous layer to a set of neurons that produce the final output. Adding a Sigmoid activation layer maps the output to a range between 0 and 1, which can lead to a better performance and higher accuracy. Finally, another dense layer was used to output of the final predictions.

The key advantages of using a CNN over other methods is that it has the ability to learn meaningful features from raw data, making it a robust to small distortions. A CNN can be trained on large, complex data sets learning nonlinear patterns between the input data and the output labels.

Hyper-parameter tuning

The hyperparameters that can be adjusted in our CNN are the learning rate, number of epochs and batch size.

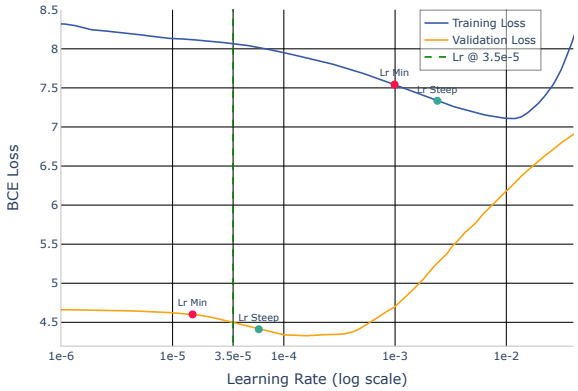

Fig. 4: Graph comparing different learning rates and losses for the validation set and training set respectively, with the chosen learning rate of 3.5e-5 also displayed.

Figure 4 depicts two different curves for the validation set and training set, with the y axis representing the Binary Cross Entropy Loss and the x axis representing the Learning Rate. The process of choosing an appropriate learning rate can be simplified, as described by Smith [9] and Gugger [10]. A custom keras callback was employed to vary the learning rate, allowing for the determination of a good starting learning rate. In addition, the ”Lr min” and ”Lr steep” values were calculated for both curves. The ”Lr min” represents the learning rate that results in the minimum loss value, which is then divided by 10. The ”Lr steep” corresponds to the learning rate that produces the steepest negative gradient for the loss value. It is noteworthy that the chosen learning rate falls between the validation set’s ”Lr min” and ”Lr steep” rather than between the training set’s ”Lr min” and ”Lr steep.” This is because the model’s generalization ability is better evaluated using the validation set, and selecting a learning rate in the range of the validation set’s ”Lr min” and ”Lr steep” ensures a balance between performance on the training and validation sets. It is also important to note that the learning rate values of the training set’s ”Lr min” and ”Lr steep” may not be suitable choices because they correspond to values on the validation set’s loss that are ascending. Choosing a learning rate within this range would result in a rising validation loss, indicating that the model is overfitting to the training set and is not able to generalize well to unseen data. Hence, the learning rate is chosen from the range of the validation set’s ”Lr min” and ”Lr steep” to ensure that the model performs well on both the training and validation sets and has good generalization ability.

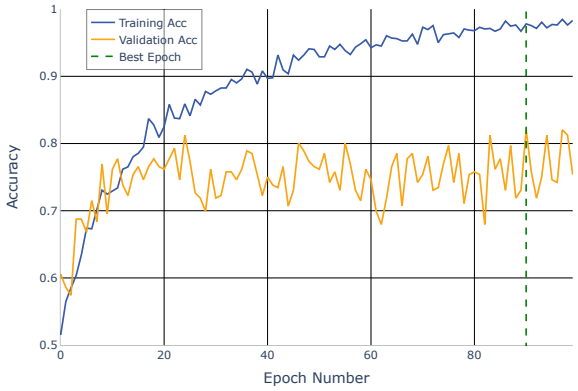

Fig. 5: Graph showing training and validation set accuracy, when varying the number of epochs.

Fig. 6: Graph showing training and validation set BCE loss, when varying number of epochs.

The trade off for more epochs is an increase in computational time, however more epochs results in a higher accuracy. The effects of varying the number of epochs can be seen in Figure 5, which shows the training and validation accuracy of

the model. The training accuracy converges to 100% whereas the validation accuracy is inconsistent throughout, due to the small size and low quality of the dataset. Figure 6 shows the decrease in BCE loss as epochs increases. It was decided to use a value of 85 for the number of epochs, as this allows the model to train on a large amount of data and returns a low BCE loss with a validation accuracy of 82%.

A study by Bengio [11] found that a batch size of 32 was sufficient for a smaller dataset, and although a larger batch size increases computational efficiency, the trade-off is that the weights are updated less frequently.

To maximise the performance of the CNN, the in-built optimizer Adam was applied to the CNN. Adam uses AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. This is particularly useful as the dataset used in this report contain images that are not consistent in camera angle, clarity and solar panel location.

C. Neural Architecture Search Network with SVM

The CNN model was further developed by the use of transfer learning [12] - specifically the Keras NASNet (Neural Architecture Search Network) deep learning model which automatically identifies the best neural network architecture for a given problem. The goal of NASNet is to provide cutting-edge performance on a range of computer vision tasks, including semantic segmentation, object identification, and image classification. An SVM (Support Vector Machine) is a ML algorithm that attempts to find a hyperplane that separates the data into different classes with the maximum margin between them. The algorithm penalizes classifications more severely by squaring the difference between the predicted score and the true label. Adding SVM to NASNet should aid in distinguishing the binary classification of each image when training our model.

Hyper-parameter tuning

To process the images for training, the Keras ImageData-Generator function was used to reshape the data to 224x224. This was found to be a good size to retain information whilst not overloading the computation.

As with the original CNN, the hyper-parameters that can be adjusted are the learning rate, epochs and batch size. From this, the step size of the epochs was determined by dividing the size of the data-set by the batch size using integer division. For the dataset in this report, the step size was calculated to be 35 for training and 8 for validation. All layers in the model, except the last 5, were set to untrainable such that the model could be generalised for transfer learning.

Due to the computational strain on loading and training, the model was not flexible to fine tuning the hyper-parameters. Therefore the same optimized values from the CNN were inputted including the in-built Adam optimizer, and a comparison of accuracy over 100 epochs was computed.

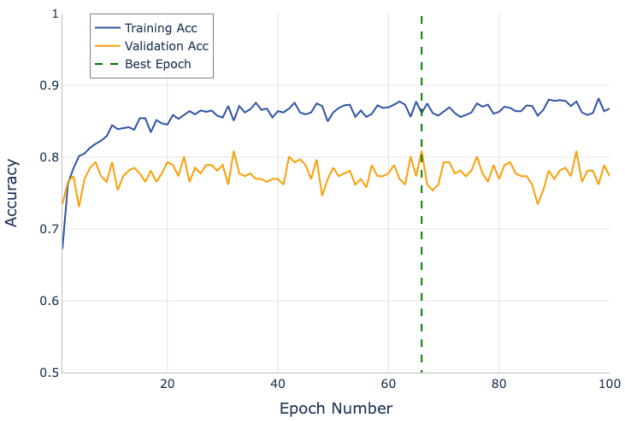

Fig. 7: Graph showing training and validation set accuracy, when varying the number of epochs.

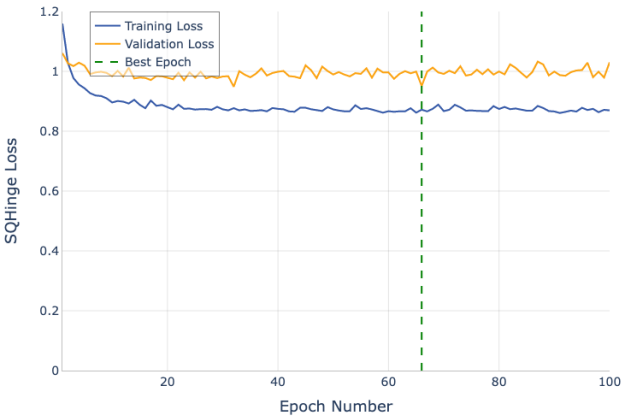

Fig. 8: Graph showing training and validation set SQHinge loss, when varying number of epochs.

D. Object Detection

The next stage of this report was to explore an object detection model that can identify and localize multiple objects in a frame of a video. It was decided to experiment with the You Only Look Once (YOLO) single stage object detection model. Single stage models favor processing time over accuracy in comparison to a two stage model, which separately detects objects and then classifies them. The YOLO object detection model was chosen for this report due to its fast processing time and ability to detect and localize multiple objects in an image, both of which are necessary for use with drone footage.

The YOLO model divides an image into a grid of cells, and predicts bounding boxes around the coordinates of the object's location in the image. Each box is annotated with a probability of the object belonging to a certain class.

YOLO uses a convolutional neural network to predict the bounding boxes and object probabilities. The network is trained on a large dataset of annotated images using a loss function that penalizes incorrect predictions. The pre-trained weights of the network can then be fine-tuned on a smaller data set of images that are annotated with the location of the solar panel and its class.

V. Experimentation & Analysis

To effectively compare each classification model's performance, consistent metrics must be chosen. As such, accuracy and F scores were chosen as suitable measures for how well the method classifies. Accuracy, calculated using equation (2), simply measures the ratio of correctly classified data points to the total number of cases examined.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

The F score, calculated using equation (3), is a measure of both precision and recall. The variable β describes the weighting of precision to recall; recall is considered β times as important as precision. In this report, the F1-score ($\beta = 1$) and additionally the F2-score ($\beta = 2$) were chosen. The F2-score was included as it was decided that it was more important to correctly classify as many positive samples as possible, thus recall is more important.

$$F = \frac{(1 + \beta^2) \cdot (precision \cdot recall)}{\beta^2 \cdot precision + recall} \tag{3}$$

Where *precision* and *recall* are defined as:

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

Model	F1-Score	F2-Score	Accuracy
KNN	0.555	0.476	0.647
CNN	0.764	0.808	0.771
SVM-CNN	0.489	0.489	0.787

TABLE I: Scores of the different models

For the YOLO model the mean average precision (mAP) score [13] can be used instead of F score. The F score is a single value that represents the mean of precision and recall therefore it does not reflect the overall performance of the model at different levels of confidence. The mAP metric can be calculated using equation (4) measuring accuracy and completeness of the models boundary box detections. A higher mAP score would mean the model can accurately detect objects with a high confidence level.

$$mAP = \frac{1}{n} \sum_{i=1}^{n} precision \tag{4}$$

Set	mAP Score
Training	0.281
Validation	0.195
Test	0.220

TABLE II: mAP scores of YOLO

VI. Discussion

The data-set used in this report incorporates imperfect photographs with a variety of camera angles and distances. This is important when training the models, as footage from drones and other pertinent equipment will not always be in the optimal camera positions.

Binary Classifiers: A variety of binary classifiers have been explored with varying successes. Table I shows that the SVM-CNN had the highest accuracy out of the binary classifiers, correctly predicting 78.8% of images in the test set. This far outperformed the KNN, which only had a slightly better performance than the expected accuracy of randomly guessing the class of each image. This is unsurprising, as the KNN only uses raw pixel values and does not perform any actual learning, where as the layers of the CNN and SVM-CNN make the model much more robust against noise. Although the CNN had a lower accuracy then the SVM-CNN, it had a significantly higher F2-score. As it is more important to correctly identify dirty panels than it is to falsely flag dirty panels, this could indicate that the base CNN would be more suitable for use in the field.

Object Detection Model: The YOLO model was evaluated by the mAP, and the model's performance can be seen in Table II. The mAP score for the test set is greater than that for the validation set, indicating that the model is capable of generalizing well to new, unseen data. However, the difference between the mAP scores of the Test and Training sets may suggest that the model has slightly overfitted on the Training data. Overall, the mAP scores suggest that the model needs further improvements to perform better on the Test set and make more accurate predictions on new data.

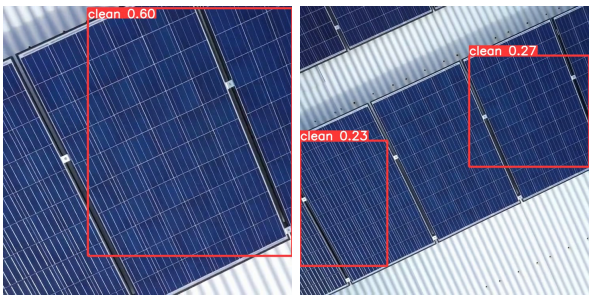

Fig. 9: Object detection from varying distances

A limitation of the YOLO model is shown in Figure 9. The boundary boxes can only be drawn vertically and horizontally, which is not necessarily consistent with the orientation of the panel in the image. Additionally, the distance of the camera from the image was found to affect the confidence level. The greater the distance, the weaker the confidence.

Fig. 10: Object detection from varying angles

Different perspectives can also impact the model's performance. This is likely due to the aforementioned limitation, where lower camera angles can result in incorrect classifications, as shown by the left image in Figure 10. The image on the right (Figure 10), shows that a birds-eye view significantly improves accuracy of classification and boundary box position.

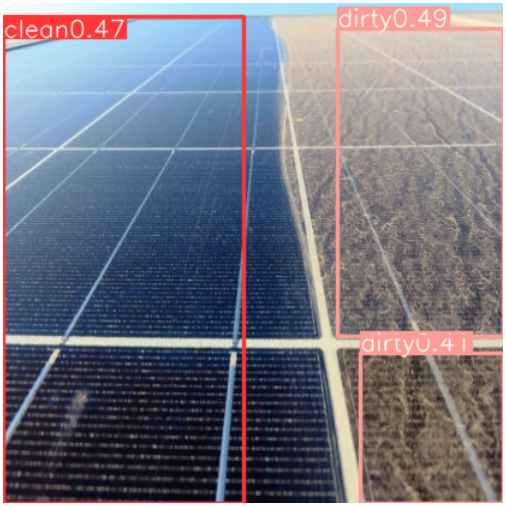

Fig. 11: Object detection of both clean and dirty in same image

Figure 11 depicts an image of a clean and a dirty solar panel in the same frame, with respective labelled bounding boxes. Notably, the CNN classified the entire image as clean with 100% confidence, failing to differentiate between the two solar panels, whereas the YOLO model accurately recognized and labelled the clean and dirty areas with labelled bounding boxes. This result demonstrates the importance of utilizing object detection models over image classification models, especially in potential applications such as drone-based monitoring of solar farms, where detection of even a small number of dirty solar panels can significantly affect the overall energy output. The capability of YOLO to identify and track individual objects in an image makes it a promising tool in such applications. An example of the YOLO model being used on drone footage can be seen via [14].

VII. Conclusion

Each model has attempted to build upon the short comings of the previous, leading to the final YOLO model. It is evident that this final model, when adequately trained, is far more successful in achieving the project aims than the simpler CNN, as it incorporates image boundaries along with confidence levels. This is an important distinction between the previous models as it highlights the relevant areas of an image and accounts for varying levels of 'dirty' as apposed to a binary classification. The real-time processing of the YOLO model makes identifying dirty solar panels a potentially fully automated task, subsequently reducing maintenance costs.

VIII. Future Work

To improve upon the research thus far, a larger and more precise dataset would allow for more reliable conclusions to be found. An extension to the final YOLO model could be the use of a kernelised correlation filters tracker [15], which would speed up the process of labelling the data. Another model that could be explored is an unsupervised learning method, which could detect dirty images from any given input.

References

[1] Shaharin Anwar Sulaiman, Atul Kumar Singh, Mior Maarof Mior Mokhtar, and Mohammed A Bou-Rabee. Influence of dirt accumulation on performance of pv panels. *Energy Procedia*, 50:50–56, 2014.

[2] Solar panel soiling detection using deep neural networks. *TechLabs Aachen*, 2022.

[3] Amith Khandakar, Muhammad EH Chowdhury, Monzure Khoda Kazi, Kamel Benhmed, Farid Touati, Mohammed Al-Hitmi, and Antonio Jr SP Gonzales. Machine learning based photovoltaics (pv) power prediction using different environmental parameters of qatar. *Energies*, 12(14):2782, 2019.

[4] Imad Zyout and Abdulrohman Oatawneh. Detection of pv solar panel surface defects using transfer learning of the deep convolutional neural networks. In *2020 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–4. IEEE, 2020.

[5] Kaggle solar panel dataset. https://www.kaggle.com/datasets/hemanthsai7/solar-panel-dust-detection?resource=download. Accessed 2023-20-02.

[6] Kittipong Chomboon, Pasapitch Chujai, Pongsakorn Teerarassamee, Kittisak Kerdprasop, and Nittaya Kerdprasop. An empirical study of distance metrics for k-nearest neighbor algorithm. In *Proceedings of the 3rd international conference on industrial application engineering*, volume 2, 2015.

[7] Nikolaos Kouiroukidis and Georgios Evangelidis. The effects of dimensionality curse in high dimensional knn search. In *2011 15th Panhellenic Conference on Informatics*, pages 41–45. IEEE, 2011.

[8] Vgg-16 cnn model. *geeksforgeeks*, 2023. https://www.geeksforgeeks.org/vgg-16-cnn-model/.

[9] Leslie N. Smith. No more pesky learning rate guessing games. *CoRR*, abs/1506.01186, 2015.

[10] Sylvain Gugger. How do you find a good learning rate. 2018. https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html.

[11] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade: Second Edition*, pages 437–478, 2012.

[12] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017.

[13] D Shah. Mean average precision (map) explained. *v7*, 2023.

[14] Github repository. https://github.com/EMAT31530/group-project-group-14. Accessed 2023-03-03.

[15] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596, 2014.