Scientific paper
Manuscript received February 15, 2026; revised May 14, 2026; accepted June 3, 2026

Journal of Energy

VOLUME 75 Number 2 | 2026

journal homepage: http://journalofenergy.com/

Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method

Ganghee Jang, Cameron Robinson

***Summary* — In general, solar power is generated by solar panels, but their efficiency depends on panel conditions, which are affected by environmental factors. To consider the variation, we need a smart cleaning mechanism. In this work, we used a data-driven approach to develop a control algorithm that accounts for the wide range of dirty-surface conditions on solar panels. As another essential design space was the budget, various feature development methods were considered to identify more power- and performance-efficient approaches. Two cleaning systems — a wiper-based and a rail-based system — were designed and used to evaluate the developed algorithms. In our experiment, the wiper-based system outperformed the rail-based system in both cleaning speed and power efficiency. Algorithms based on hand-engineered features can reduce algorithmic size and inference time while maintaining competitive performance. However, the algorithms with learned features showed consistently decent performance and development time.**

***Keywords* — Solar panel, cleaning, data-driven control algorithm, hand-engineered features, dark channel, dataset.**

Fig. 1. Solar panel arrays covered with snow.

I. Introduction

Solar panels are a popular renewable energy source. However, the performance of each solar-powered system can be affected by the condition of its solar panels' surfaces in many ways. If the solar panel system has a goal of making a profit, it can be severely hindered by poor weather conditions. Power generation can also be interrupted when the surfaces of the solar panels is covered. For example, sandstorms in the Middle East can reduce power generation efficiency or even cause total power loss, resulting in financial losses for power plants, as studied in [1].

Sometimes, solar panels can get damaged by dirt. Uneven heating from sticky materials, such as organic waste, moss, etc., can cause damage to the panels over time [2]. Therefore, the timing of cleaning is essential. Dirty panels should be cleaned before the dirt has solidified. In the case of light snow, it would be better to clean the surface when the snow stops. However, as in the example shown in Fig. 1, cleaning a snow pile requires significant effort, and sometimes even snow removal by hand. If cleaning were performed while it was snowing, the solar panels would be ready for power generation right after the snow stopped.

The decision to clean and the timing of the action can vary depending on the type of dirt and weather conditions. Also, as climates differ by region, dirt types and weather will vary. So, data-driven algorithm development is a necessary design direction.

Running algorithms from the data-driven approaches tends to require more computing power, however, power and performance efficiency for the target embedded system should be considered. We approached our designs from two different feature development methods: hand-engineered, and automatically generated features.

To test the cleaning algorithm, we developed fixed types of cleaning mechanisms for the design exploration. There has been related research activities in the domain of fixed cleaning mechanisms for solar panels. Still, only a small amount of attention has been given to the development of controller algorithms. On-time cleaning requires smart identification of dirt species - soil, sand, snow, leaves, etc.

The contribution of our work is as follows:

(Corresponding author: Ganghee Jang)

Ganghee Jang and Cameron Robinson are with the Oregon Institute of Technology, Klamath Falls, OR, USA (email: ganghee.jang@oit.edu; robinson.cameron75@gmail.com)

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

- Development of the training dataset for the development of a control algorithm using a data-driven approach. The dataset development reflects local weather conditions.
- Exploration of design space for realistic cleaning workloads. For this purpose, we design cleaning test procedures to numerically compare two different cleaning systems.
- Exploration of algorithm design spaces for better performance and power efficiency.
- Development of two different solar panel cleaning systems and comparison of their performance using the test procedures.

In Chapter II, we discuss similar research to contrast the different approaches we took. Then, we explain our approaches — dataset, hardware cleaning mechanisms, and dirt detection algorithms — in Chapter III. The performance measures for the hardware mechanisms and algorithms are listed in Chapter IV. Lastly, the conclusion and future work are discussed in Chapter V.

II. Related Works

Renewable energy investment has increased, especially in photovoltaic electricity, which was expected to account for about 20% of global energy demand by 2050 [1]. Given the importance of photovoltaic (PV) electricity, the cleaning and protection of solar panels has attracted researchers' attention.

It is obvious that dust accumulated on solar panels would reduce the amount of solar energy absorbed and converted into electricity. Energy lost from dust or dirt was reported between 7% and 50% [3]. If cleaning is necessary, the next question is how we can apply an efficient cleaning methodology.

In [1], the authors classified commonly-used mechanisms into Brush Cleaning System (BCS), Electrostatic Cleaning System (ECS), Heliotex Cleaning System (HCS), Robotic Cleaning System (RCS), and Coating Cleaning System (CCS). The classification is somewhat uncertain; for example, RCS can also have brushes to clean dust. Also, soiling can be prevented with immediate cleaning. In other words, specific cleaning methods depend on the chosen cleaning strategy. BCS systems are the least expensive in terms of hardware, so they are the most cost-efficient among the systems categorized in [1].

Many BCS systems have been suggested. One of the most common choices is to move a wiper with a silicone brush or roller in a single direction on the rails [2], [4], [5], or a two-dimensional rail system to clean multiple solar panels aligned on the same rails [6]. Using two sets of rails has clear benefits over a single-rail cleaning system. In [6], the y-axis movement wipes down a single solar panel while the xaxis rails move the wiper to the neighboring solar panels. The two-axis implementation was utilized to expand the cleaning system to cover an array of solar panels. A single motor is used for each axis, and rollers are used to slide the wiper. Rollers are kept in the E-shaped channel to hold the brush against the solar panel securely. This system lacks intelligent dirt detection, so soil can be compacted on the surface. Some systems have water sprinklers to handle this issue.

The previous works listed above failed to use standardized metrics or common standards to compare with other designs. Habib et al. [5] compared many different blade materials with the amount of scratches after cleaning with varying cleaning loads. Still, the discussion of the definition of the dirt species was weak, and no numerical measure of the amount of sand used in the experiments was provided.

To trigger the cleaning action on time, dirt detection algorithms and mechanisms are required. The algorithm depends on the modality of the input data or characteristics; the choice of sensors is critical for the design of dirt-detection mechanisms.

Between the choices, non-imaging sensors would be less complex than imaging sensors in terms of data and computational complexity. Manish's work [6] is one of the simplest yet efficient approaches. The author used an infrared sensor to determine day/night shifting, overheating, etc., and to make cleaning decisions together with temperature and powergeneration measures. However, power generation depends on seasons and weather conditions, and this work did not fully account for these performance sensitivities. Habib et al. [5] addressed this issue by adding an LDR (light-dependent resistor) to determine whether dust covers the panels. Still, coverage of variations was minimal, as only sand was used to determine the threshold value for triggering cleaning.

To consider seasonal variation, Rupin, et al. [2] used a temperature sensor while [4] utilized a color sensor, a humidity sensor, and a temperature sensor. However, thresholding was still used to trigger actions without discussion of long-term performance. This solution mainly focuses on soil dirt on the solar panel, which turns the panel's color gray. Cleaning was performed based on a threshold of color sensor values. However, this method cannot handle many different kinds of dirt or dirt spots. Rupin et al. [2] employed similar rail-based mechanisms, with a temperature sensor and a humidity sensor, to detect dirt on the solar panel surface. This is one of the rare works that measured cleaning time as a performance metric. It took 300 seconds to finish a single cleaning process, moving the wiper from one side to the other. A spiral roller brush improves cleaning efficiency.

To handle wide variety of dirt species and seasonal change, data-driven algorithm development should be considered. Deep Neural Networks (DNNs) are technically mature and widely adopted, so if we have a proper dataset, developing a high-performing algorithm is easily achievable, even by a hobbyist or blogger [7]. Convolutional Neural Networks (CNN) [8] have demonstrated their effectiveness for identification and classification by automatically learning features. However, algorithms using CNNs require higher computational complexity than the methods we discussed above, so we should carefully assess their usefulness when designing systems under a tight power and cost budget. For example, Hu et al. [9] used an imaging modality to detect dirt on solar panels, but they handengineered their algorithms using image processing methods such as histogram equalization, gray-level concentration, edge detection, etc., rather than using more automated features with CNNs.

We can probably use image processing operators to generate features, and then apply clustering or classification using supervised machine learning methods such as Support Vector Machine (SVM) [10]. These methods will be more timeefficient during development than fully hand-engineered algorithms when working with complex multidimensional data. At this point, we had a design question whether handengineered features would work better than learned or automatically generated features. In terms of development or design time, learned features such as convolutional layers of CNNs will be more efficient. However, in terms of performance, many researchers are still debating the comparison between the two approaches, as in [11], [12]. In the literature, authors agreed that methods using learned features have been outperformed in many domains; however, handengineered features can improve performance when combined with learned features.

For high-computational-complexity data, hand-engineered approaches had an edge over methods with learned features, but were later surpassed, as in the example of [13]. Therefore, there may be some domains where hand-engineered approaches are more efficient.

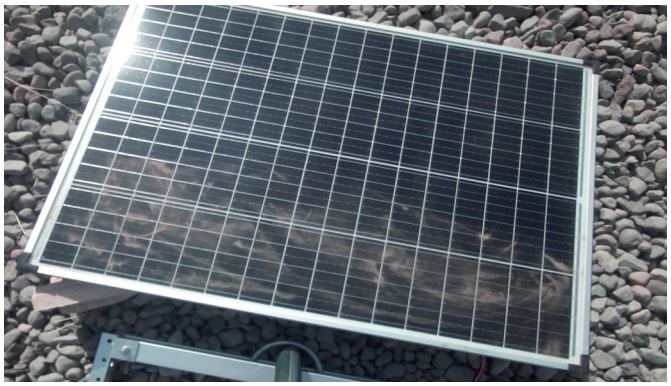
Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

III. Suggested Methods

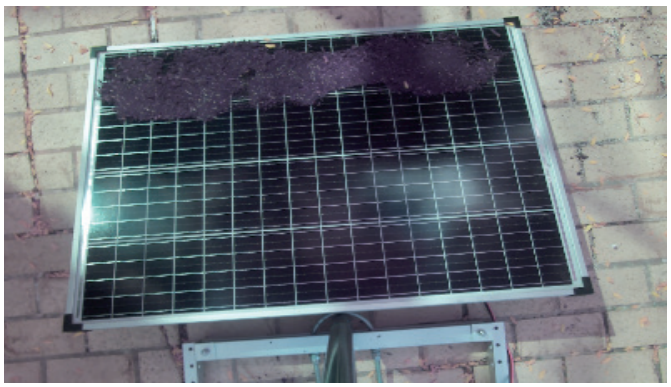
A. Dataset development for a smart control algorithm

The system's intelligence should stem from the careful design of a dataset, as unseen events are not easily handled in data-driven algorithm development. Therefore, developing a dataset that reflects the weather conditions at the target installation site is essential for comparing the efficiency of cleaning mechanisms.

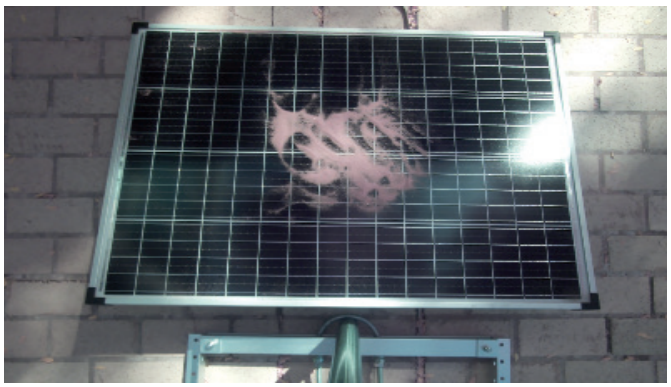
Each dirt species was selected based on Oregon's weather. The types of species are soil, sand, and leaves. For the soil, we used garden topsoil, and for the leaves, we used a collection of dried leaf litter. We used two different types of sand: orange sand with medium-sized granules and fine brown dust we sifted from sand and small rocks. Other variations we added include camera angle, camera distance, date, and time. Shade, dry, and wet conditions of the dirt species were applied for further variation. With the type of species, the ground truth also carries information about dirty (1) and clean (0). The areas of dirt within the three-by-three division of the solar panel were encoded in the image name, as shown in Fig. 2. For example, the image of Fig. 2 (a) has a file name with a label, 000-000-111, which means the bottom row is covered with the dirt specie.



(a) Sample Dust Image - Label: 000-000-111



(b) Sample Soil Image - Label: 111-000-000



(c) Sample Red Sand Image - Label: 000-010-000

Fig. 2. Sample images from solar panel training dataset. labeled 0 – 8 from left-to-right top-to-bottom

TABLE I
Image Dataset Species Count

Species	Image Count
Soil	96
Sand	200
Leaves	180
Clean	64

The three images show the camera angle variation (max skewed position, slight variation, and normal position from top to bottom), dirt specie variation (dust, soil, and red sand), and covered area variation. Data collection was conducted during the summer, and solar panel mechanism tests were conducted during the winter. A layer of snow was tested during the cleaning hardware tests, but was neither included in the dataset nor used during the performance evaluation of the trained model.

With the setup, we collected up to 540 images, and the detailed species and counts are listed in Table I. One thing to note is that we have 246 images out of 540 with the extra label "shade" to check whether the trained algorithm can distinguish between shaded and dusty surfaces.

The images were collected using a Raspberry Pi 4B and a Raspberry Pi Camera Module 3 with a 120-degree lens. An image has a resolution of 4096 x 2592 and is saved in the TIFF format. The data collection system was reused as the controller for the cleaning mechanisms.

B. Cleaning mechanisms: wiper system and rail system

As discussed in Chapter II, the cleaning system without a dirt detection mechanism should be run periodically. This type of solution can cause two issues. The system can clean even when there is little dust on the panel, resulting in wasted power. On the other hand, dust piled on the panel can solidify, requiring more cleaning power or even additional mechanisms, such as water spray, as discussed in [6]. With a dirt-detection mechanism, the cleaning system can be more power-efficient and designed with less powerful motors.

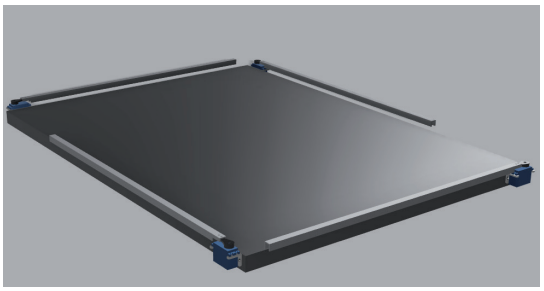
Here, the question is how to evaluate whether the cleaning system has adequate power. To answer the question, we designed two different cleaning systems with power measurement circuitry (Fig. 7). The power measurement for system power consumption compares the power used by the two different cleaning mechanisms we developed. Even though we added an additional circuit for monitoring panel power generation, the study on optimization of current power generation is left as future work.

The two cleaning systems developed are a wiper system and a single-axis linear rail system. Both systems used cleaning arms fitted with rubber windshield wiper blades. Each system was installed on a 100 W single-panel monocrystalline solar panel. The dimensions of the solar panels are 91.4 cm x 68.6 cm with a thickness of 3.56 cm.

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

For the wiper system, we used four 61 cm cleaning arms, each attached to a corner of the solar panel and operated by a servomotor, as shown in Fig. 3. Each arm can move 90 degrees clockwise and counter-clockwise. We ordered arm movement to ensure the best cleaning efficiency. A single run takes 15.3 seconds on average. To ensure that the wipers did not get in each other's way when cleaning, they were run in the following order (relative to Fig. 3.(a)):

1. Right motor rotates 90-degrees clockwise (CW)
2. Front motor rotates 90-degrees CW
3. Left motor rotates 90-degrees CW and then 90-degrees counter-clockwise (CCW)
4. Front motor rotates 90-degrees CCW
5. Right motor rotates 90-degrees CCW
6. Left motor rotates 90-degrees CW
7. Back motor rotates 90-degrees CW and then 90-degreesCCW
8. Left motor rotates 90-degrees CCW



(a) Wiper System - Angled

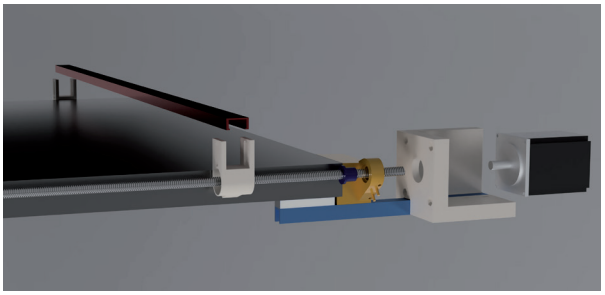
(b) Wiper System - Top Down

Fig. 3. CAD Model of Rail System

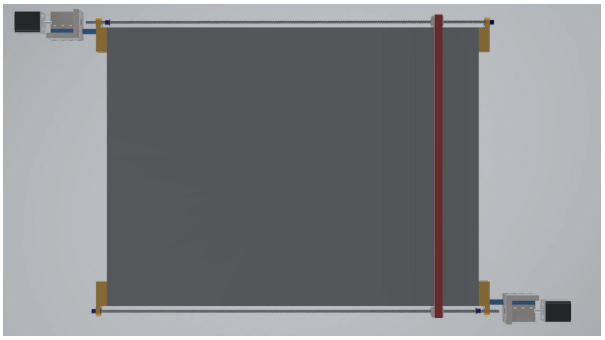
The servo motor has a maximum torque of 3.43 N·m, and all four motors are controlled by a Raspberry Pi board circuit shown in Fig. 4. The components shown in the circuit diagram, from left to right and top to bottom, are the voltage divider, current sensor, Raspberry Pi, analog-to-digital converter (ADC), solar panel, charging station, 12 V to 6.6 V buck converter, camera (not used), and four servomotors.

The rail system has one cleaning arm attached to two lead screws on each side of the solar panel, as in Fig. 5. The lead screws were rotated using two NEMA 23 stepper motors with a maximum torque of 1.85 N·m controlled by a Raspberry Pi board circuit as in Fig. 6. A single run moves the arm from one side to the other. As the lead of the screw is very dense (2 mm), a single run takes around 200 seconds.

Fig. 6 is the circuit diagram for the rail system. From left to right, top to bottom, the components shown in the diagram are the voltage divider, current sensor, Raspberry Pi, ADC, solar panel, charging station, limit switches, camera (not used), stepper motor driver, and two stepper motors. The entire system is powered by a 12 V output from the charging station. The Raspberry Pi and other



(a) Rail System - Side



(b) Rail System - Top Down

Fig. 5. CAD Model of Rail System

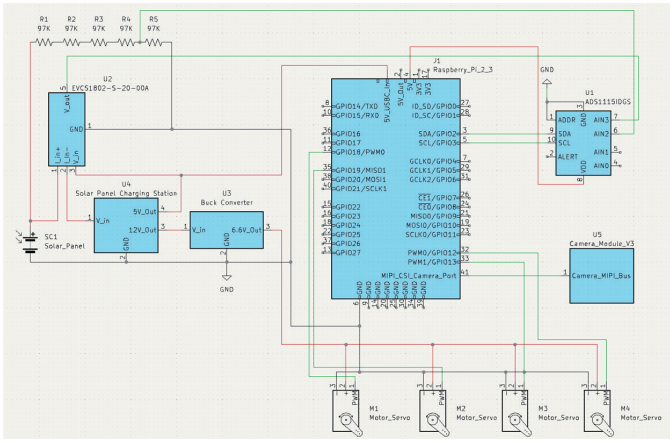


Fig. 4. Circuit for Wiper Cleaning System

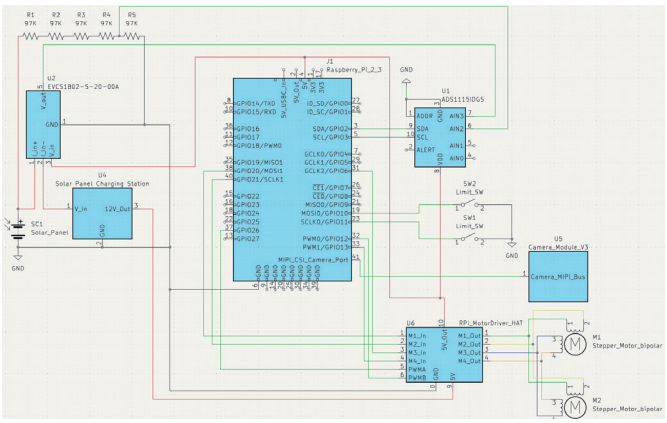


Fig. 6. Circuit for Rail Cleaning System

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

sensors are powered by 5 V, while the motors are powered by 12 V.

As mentioned earlier, to measure the system's power consumption, we connected an additional current sensor between the charging station's power output and the cleaning system, exactly as we did between the solar panel and the charging station's power input. This power measurement system is outlined in Fig. 7, where we measured and collected data from both the solar panel and cleaning systems. The power measured from the panel was power generated by the system, while the power measured from the cleaning systems was power consumed by the system.

C. Data-driven controller algorithms

The developed cleaning mechanisms could clean every dirty solar panel setting (we will discuss further hardware performance in the next chapter) with fixed counts of run(s); we did not utilize dirt species labels for the algorithm development. The developed algorithms are to identify dirty solar panels and areas with dirt to evaluate their dirt condition. The overall pipeline of the dirt detection mechanism is explained in Fig. 8.

To explore the design space of power and performance efficiency, we approached algorithm development from two different perspectives. We tried both hand-engineered and automatically generated features for algorithm development. In general, automatically generated features take less time to develop and, in many cases, outperform hand-engineered features due to advances in technologies such as hardware accelerators. However, hand-engineered features can yield more power-efficient algorithms when developers correctly interpret the problem.

When developing our models, we focused on two main types of models: single-label binary classification models that indicate whether the entire panel is clean or dirty, and multilabel binary classification models that classify individual 3x3 grid segments of the panel. These are the same segments as illustrated in Fig. 2.

Fig. 8 is a system block diagram showing the integration of the cleaning algorithms with our cleaning mechanisms.

1) Approach using hand-engineered features: As we can see in Fig. 2, from some distance, dust looks similar to the hazy or blurry effect on images. This observation drove us to start with hand-engineered features from haze detection and degree estimation [14].

$$I^{dark} = \min_{(i,j)\in\Omega(x,y)} \left(\min_{c\in r,g,b} \frac{I^c(i,j)}{A^c} \right) \quad (1)$$

$$C_{RMS} = \sqrt{\frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I_{ij} - \bar{I})^2} \quad (2)$$

Dark channel can be expressed in a mathematical formula as in Equation 1. $\frac{I^c(i,j)}{A^c}$ means the removal of atmospheric light A^c from the original image I^c. Then, select block-wise (patchwise) minimum. Another feature set, RMS contrast values can be calculated with Equation 2. The indices, *i* and *j* are to indicate a pixel, and I_{ij} is the intensity of the pixel pointed by (*i,j*), and *I* is the average intensity value in a patch.

We followed the empirical approach to select dark channel values, as suggested, and used the same method for patch-wise contrast values.

In both cases, the RMS contrast or dark channel was calculated for each pixel in the 256x400 BGR images, which were then flattened and sorted (after removing the now duplicate second and third image channels). We then extracted values from each vector at select indices. The dark channel values were selected as a part of the feature set at 50%, 25%, 10%, 5%, and 1% of the overall length of the vector from the whole patches (e.g., if the overall dark channel vector was 100 elements, then the 50th element was selected for the 50% value). The RMS contrast features were selected from the shorted patch-wise values at the 100%, 95%, 90%, 85%, 50%, and 10% locations.

The 11 selected features were used to train algorithms using a support vector machine (SVM). We set the regularization parameter to 900 and the gamma to 12.0, and used a polynomial kernel of degree 3 during training, as these values worked best in our experiment. The SVMs were trained using k-fold cross-validation on

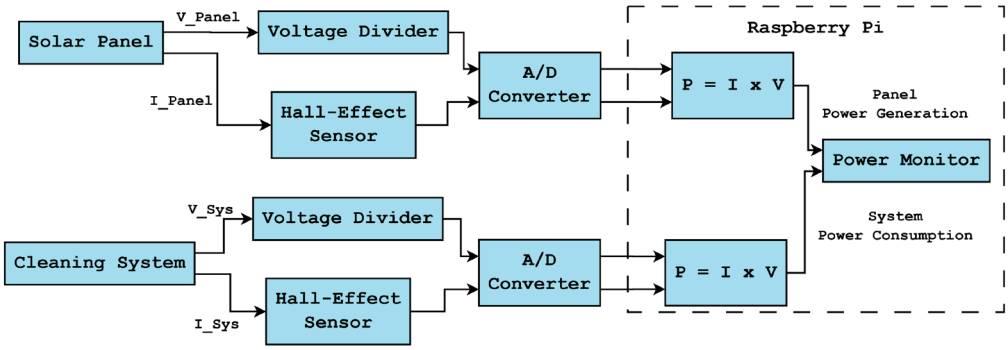


Fig. 7. Power Measurement Block Diagram

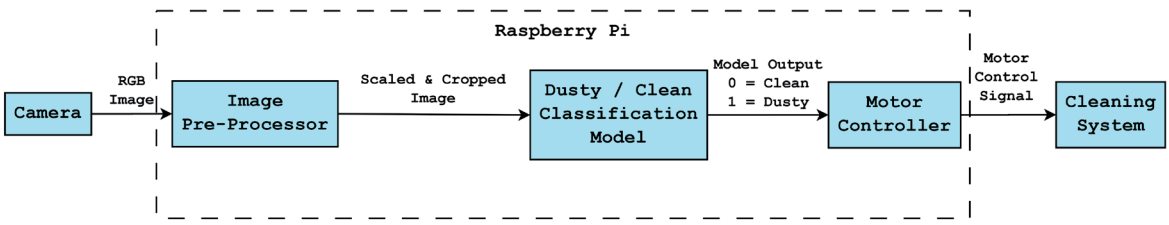


Fig. 8. Cleaning Decision Block Diagram

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

80% of the dataset, while the other 20% was used for the final test.

For training the feature-based deep-learning models, we split the original dataset, using 80% for training, 10% for validation, and 10% for testing. We applied random brightness adjustments to the images in the training set. In total, we had 54 images in the test and validation datasets, and 880 images in the training dataset. At runtime, the input images were further split into nine segments and individually passed into the models for inferences. Unweighted binary-cross-entropy (BCE) loss, like that in Equation 3, was used for training.

2) Approach using automatically generated features: When we began training the deep-learning models, we had no prior knowledge of the newly developed dataset. Therefore, we selected various neural networks of different sizes.

Our architecture exploration was focused on convolutional neural networks (CNNs). CNNs are well-known for their high performance, thanks to automatically generated features from convolutional layers. At one extreme of this class of neural network architectures, there are Fully Convolutional Networks (FCNs), such as U-Net [15]. We can interpret the architecture as a combination of an autoencoder and a decoder [16], with skip connections between the corresponding pairs of the encoder and decoder in a U-shaped architecture. This design enables the model to efficiently find a feature map without using fully connected layers, thereby reducing memory usage significantly. Since we need a prediction about the presence of dirt, fully connected layers were added to flatten the feature map and convert it into a probability of dirt presence. From the experiment, four layers for both the down- and up-sampling performed well. The overall model architecture is in Fig. 10; the larger model is for the 3x3 segment model we trained while the smaller version is for the binary model.

ResNet [17] used the skip connection in a different way to avoid degradation in deeper networks. The skip connection is in the residual building block in the forward direction. We used ResNet50 and ResNet101 as the baseline and modified the fully connected layer to get output logits indicating the probability of dirt on the panel (nine logits for multiclass, one for binary). The overall model architecture is depicted in Fig. 9.

Lastly, we trained customized LeNet [8] based models, such as in Fig. 11. We prepared three other size variations from the CNN1 model shown in the figure. The size variations mostly came from fully-connected layer blocks. The fully-connected (FC) layers' input and output sizes of CNN2 are 51200 in, 13 out; 13 in, 9 out; and 9 in, 9 out. The FC layers' sizes for CNN3 are 12800 in, 9 out; 9 in, 9 out; and 9 in, 9 out. Lastly, the FC layers' sizes for CNN4 are 12800 in, 8 out; 8 in, 9 out; and 9 in, 9 out. These are all for the first, second, and third FC layers, respectively. CNN2 also uses a 2x2 average pool kernel, while CNN3 and CNN4 use 4x4 kernels. To create a model smaller than CNN4 we decided to remove all but one fully-connected layer resulting in the model illustrated in Fig. 12.

The binary and 3x3 segment models were trained on 256x400 BGR images (resized and cropped from the original dataset). Since we wanted the model to learn to segment the panel relative to the image, we minimized augmenting the dataset with methods that changed the panel's orientation within the image, such as reflections and rotations. Instead, we used techniques such as brightness and contrast adjustments, region segmentation, blur, and edge segmentation, but ultimately decided to train the final models with random brightness adjustments and some random Gaussian blur.

We split the original dataset, using 80% for training, 10% for validation, and 10% for testing. Augmentations were applied to the training data after the split, which resulted in 54 images in both the validation and test sets, and 5,616 images in the training dataset.

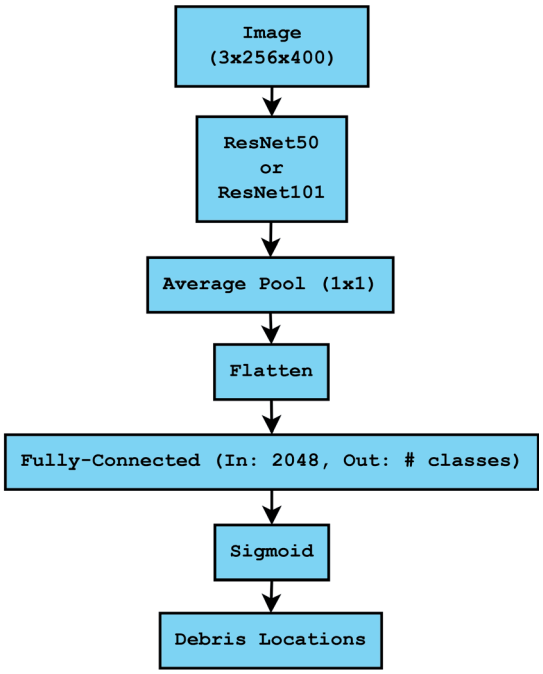


Fig. 9. ResNet Model Diagram

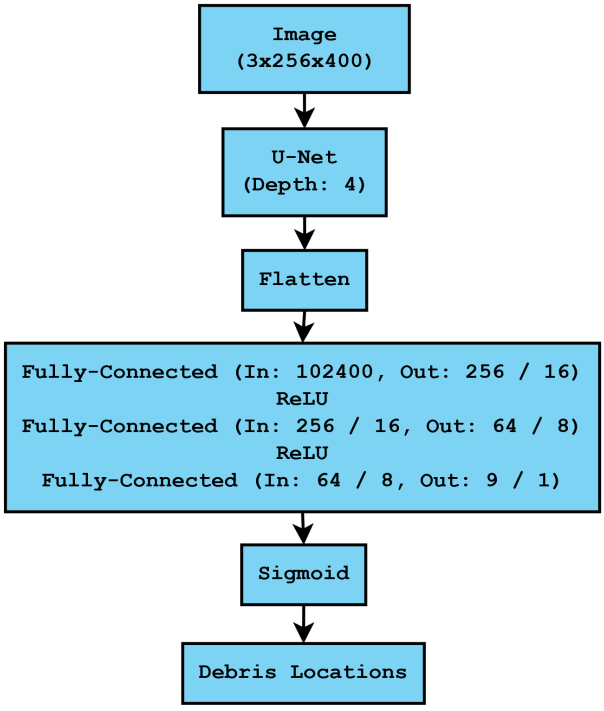


Fig. 10. U-Net Model Diagram

Equation 3 is the loss function used to train the deep learning models. When training binary models, we did not weight either class, but when training the 3x3 segment models, we added more weight to the positive class (dirty).

In the equation, W_p is the weight for the positive class, while W_n is the weight for the negative class. y_n and p_n are the labels and predictions for each panel section, respectively. The sum of the loss for each segment is averaged to get the total loss for the input image.

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

$$BCE\ Loss = \frac{1}{N}\left(\sum_{n=1}^{N} -\left(W_p\, y_n \ln\left(p_n\right) + W_n\left(1-y_n\right)\ln\left(1-p_n\right)\right)\right) \quad (3)$$

We used PyTorch [18] for the algorithm development. The system specification of the machine used for the model development is as follows:

- CPU: Intel(R) Core(TM) Ultra 9 185H 2.3 GHz
- GPU: NVIDIA RTX 2000 Ada Generation Laptop GPU (8 GB Dedicated, 16 GB Shared)
- SSD: BG6 KIOXIA 1014 GB
- RAM: 32 GB 5600 MT/s

We further explored model compaction using optimization methods of ONNX [19] and pruning [20] to optimize for the target embedded system. Nine variations were created from the trained models discussed above. The variations were from combinations of four different operations: converting the PyTorch model to ONNX, pruning the PyTorch model, preprocessing the ONNX model, and performing static quantization on the ONNX model.

The pruning method is an unstructured pruning [21], which sets 20% of the lowest weights over the entire trained model to zero. Preprocessing is an ONNX method intended to prepare and optimize a model for quantization by determining tensor shapes, adjusting the model computation graph, and combining model layers [22]. Lastly, quantization from 32-bit floatingpoint numbers to 8-bit signed integers is performed using ONNX static quantization optimization. We tested various combinations of these optimization methods in order to determine which resulted in a small and fast model that does not compromise too much on accuracy.

IV. Experiment Results

A. System Cleaning Mechanisms Test Results

Comparison of cleaning performance between the two systems, wiper and rail, shows steady trends. Fig. 13 compares power consumption over time between the two systems. The wiper system graph shows two runs to clean dry leaves and has a spikier graph, which explains the higher standard deviation of the wiper system results in Table II.

The total power consumption is proportional to the multiplication of average power, single run time, and count of cleaning runs (clean runs). The rail system consumes roughly six times more power for the same cleaning performance. As the rail system moves the wiper arm with two stepper motors, power consumption is less varied for different dirt species during the cleaning. The wiper system's average power consumption varies from 4.5792 W/s to 11.6426 W/s, while the rail system's average power consumption ranges from 6.0932 W/s to 6.9821 W/s.

There are two things that should be considered for the generalization of the comparisons. First, we used only soiltype dirt with moisture variations. The wiper system can consume more power than with other dirt types, such as heavier-grained or stickier dirt. Secondly, the wiper arm's moving speed on the rail system can be improved by using a longer thread distance for the lead screws. However, the power consumption rate will be different, so further experiments are required.

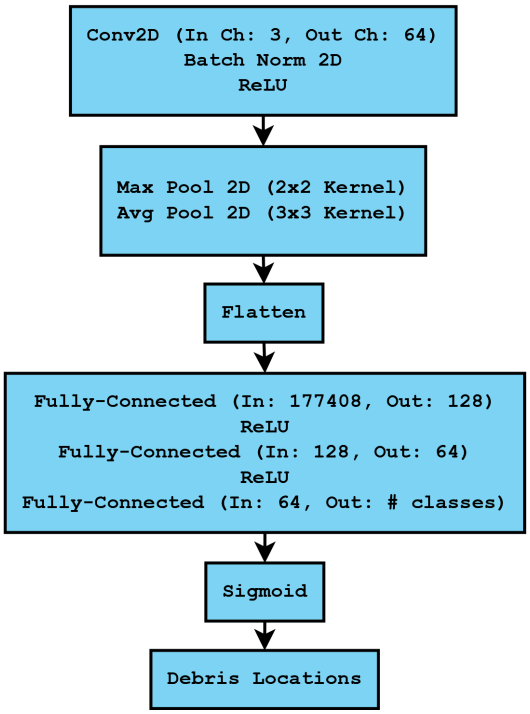


Fig. 11. CNN1 Model Diagram

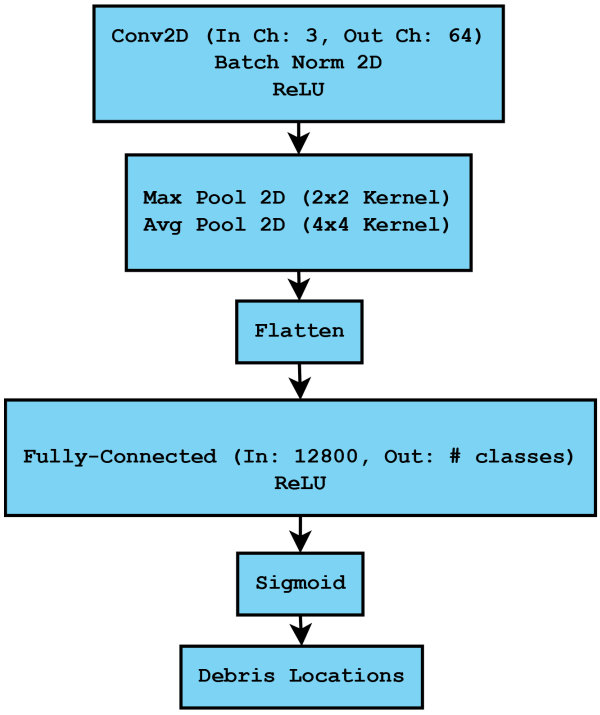


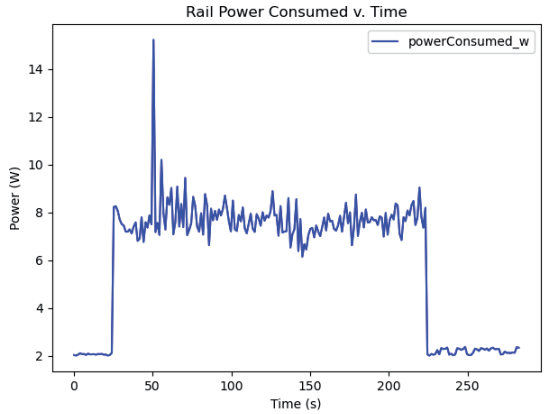
Fig. 12. CNN5 Model Diagram

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

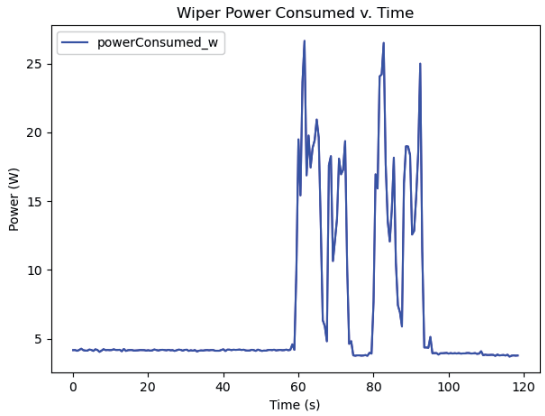
TABLE II
CLEANING SYSTEM COMPARISON RESULTS

System	Material	Avg. Power (W/s)	Power Std. Dev. (W/s)	Single run time(s)	Clean runs
Rail	Dry Dirt	6.1233	2.7381	200	1
Rail	Wet Dirt	6.0932	3.1191	199	1
Rail	Dry Leaves	6.9821	2.6381	201	1
Rail	Wet Leaves	6.1671	2.8639	200	1
Rail	Wet Leaves & Dirt	6.4994	3.0711	199	1
Rail	Snow	6.6529	2.6850	187	1
Wiper	Dry Leaves	6.6444	3.4261	15.3	2
Wiper	Dry Dirt	11.6426	4.6877	15.4	2
Wiper	Wet Dirt	4.5792	3.1269	15.4	2
Wiper	Wet Leaves	6.3422	4.6570	15.3	2
Wiper	Wet Leaves & Dirt	5.0517	3.4346	15.3	2
Wiper	Snow	5.4972	3.7457	15.3	2

Both systems are powerful enough to handle the designed dirty panel setups. The rail system needs only a single cleaning cycle, as the blade is securely down-forced, while the wiper system requires two runs of four arms to clean the remainder due to the openness at the end of each arm. Fig. 14 shows each system under cleaning action.



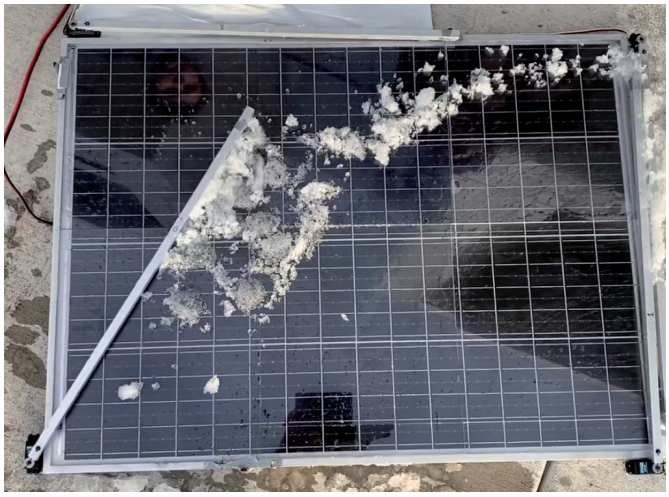
(a) Rail System Power Consumption



(b) Wiper System Power Consumption

Fig. 13. Cleaning System Power Consumption Graph with Dry Leaves

(a) Rail system in action.



(b) Wiper system in action.

Fig. 14: Rail system cleaning wet leaves, while wiper system cleaning snow.

The rail system's cleaning speed could be improved by increasing the screw leads. The one we used is a lead screw that moves the carriage 2 mm every rotation. If the rotation speed of the stepper motors is maintained, the carriage will move twice as fast with a twice-as-long leaded screw. The design update and power/performance measurements on the updated system are left for future work.

B. Binary Dust-Detection Algorithms and Model Variations

We developed and compared a variety of algorithms from two approaches: hand-engineered features and automatically generated features. The approaches from generated features, using convolutional neural networks, were developed relatively quickly and showed decent performance overall. Algorithms developed with hand-engineered features took much longer to develop and showed relatively better memory efficiency, but performance was inconsistent and sometimes much worse.

Table III summarizes the overall experiment. the first nine models in the first box are trained for binary classification: the decision whether the panel is clean (0) or dirty (1). The best performance and power efficiency was from when we used hand-engineered features. As in Table III, the binary SVM model size is about 4.5 KB, and the inference time takes just 0.0034 seconds on average on the Raspberry Pi 4B board. It was developed as described in Section III.C.1., with 11 features as explained. We will discuss hand-engineered features further in the following subsection.

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

The overall most accurate models were the ResNet models, which got 100% accuracy on the test set. U-Net, and CNNs 1 - 4 all got 98.15% accuracy. Our smallest model, CNN5, performed the worst of the binary deep neural networks with 94.44% accuracy, but is still about 50 KB, which is almost 10 times larger than SVM (binary).

We also performed tests on the impact of model compaction in Fig. 15. Static quantization had the largest effect, reducing model size to a quarter of its original size (32-bit floating point to 8-bit integer). For the larger models, static quantization also tended to significantly decrease inference time, with CNN1's inference time decreasing by 40%. On the other hand, CNN2CNN5 all showed a slight increase in inference time. For the smaller models, the most significant decrease in inference time came from converting them to ONNX variants, which sped them up by at least 50%.

Pruning did not have consistent results. In some cases, such as the base PyTorch (i.e., Torch) ResNet101, ONNX ResNet101, and Torch ResNet50, pruning decreased inference time slightly. In other cases, such as ONNX ResNet50 and Torch CNN5, pruning increased inference time.

ONNX preprocessing was similarly inconsistent. At times, preprocessing decreased the models' inference times, while at other times, it increased them. Interestingly, the difference is not consistent even between static quantization models and preprocessed static quantization models: five architectures saw a decrease in inference time, while the others saw an increase.

The best binary deep learning models depend on the application. If accuracy is more important than real-time performance, the preprocessed static-quantization ResNet50 model works well, achieving 100% accuracy with an inference time of about half a second and a memory footprint of 22.58 MB. If a balance between inference time, size, and accuracy is needed, any of the ONNX (and further) variations of CNN2-CNN4 are good options. To pick one, however, the preprocessed static quantization CNN4 strikes a good balance with about 98% accuracy, a 110 KB model size, and 12.8 ms inference time. Overall, a classifier using hand-engineered features - SVM (binary) achieved the best performance with the smallest size and the shortest inference time.

The convergence of developed algorithms is shown in Fig. 16.

C. Three by Three Zoned Algorithm Development

In a larger deployment, we can group multiple solar panels and monitor them with an imaging sensor. In this scenario, multi-zoned detection is necessary to clean only dirty panels. We performed three-by-three zoned dirt detection to explore algorithm deve-

	Torch	Pruned Torch	ONNX	Preprocessed ONNX	Static Quantization	Preprocessed Static Quantization ONNX	Pruned ONNX	Pruned Preprocessed ONNX	Pruned Static Quantization ONNX	Pruned Preprocessed Static Quantization ONNX
ResNet101	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
ResNet50	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
U-Net	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%
CNN1	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%
CNN2	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%
CNN3	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%
CNN4	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%	98.15%
CNN5	94.44%	94.44%	94.44%	94.44%	94.44%	94.44%	94.44%	94.44%	94.44%	94.44%

(a) Accuracy vs Model Variation

	Torch	Pruned Torch	ONNX	Preprocessed ONNX	Static Quantization	Preprocessed Static Quantization ONNX	Pruned ONNX	Pruned Preprocessed ONNX	Pruned Static Quantization ONNX	Pruned Preprocessed Static Quantization ONNX
ResNet101	162.74	162.73	161.98	161.99	40.84	40.84	161.98	161.99	40.84	40.84
ResNet50	89.99	89.99	89.61	89.61	22.58	22.58	89.61	89.61	22.58	22.58
U-Net	124.65	124.65	124.65	124.65	31.25	31.25	124.65	124.65	31.25	31.25
CNN1	86.67	86.67	86.67	86.67	21.67	21.67	86.67	86.67	21.67	21.67
CNN2	2.55	2.55	2.54	2.54	0.64	0.64	2.54	2.54	0.64	0.64
CNN3	0.45	0.45	0.44	0.44	0.12	0.12	0.44	0.44	0.12	0.12
CNN4	0.40	0.40	0.39	0.39	0.11	0.11	0.39	0.39	0.11	0.11
CNN5	0.05	0.05	0.05	0.05	0.02	0.02	0.05	0.05	0.02	0.02

(b) Model Size (MB) vs Model Variation

	Torch	Pruned Torch	ONNX	Preprocessed ONNX	Static Quantization	Preprocessed Static Quantization ONNX	Pruned ONNX	Pruned Preprocessed ONNX	Pruned Static Quantization ONNX	Pruned Preprocessed Static Quantization ONNX
ResNet101	2.7843	2.7805	2.1269	2.2004	0.9497	1.0916	2.2936	2.1605	1.1065	0.9925
ResNet50	1.6509	1.6292	1.0891	1.0951	0.5923	0.4901	1.1228	1.1427	0.5027	0.5212
U-Net	8.5416	9.1728	8.4413	8.9373	3.5343	4.9123	8.1550	7.3005	3.6509	4.7601
CNN1	0.3796	0.3796	0.0936	0.1027	0.0615	0.0608	0.1003	0.1116	0.0549	0.0650
CNN2	0.0317	0.0367	0.0113	0.0108	0.0134	0.0161	0.0162	0.0116	0.0182	0.0145
CNN3	0.0280	0.0268	0.0111	0.0114	0.0145	0.0139	0.0112	0.0122	0.0176	0.0142
CNN4	0.0266	0.0275	0.0114	0.0125	0.0136	0.0128	0.0117	0.0158	0.0171	0.0144
CNN5	0.0247	0.0320	0.0123	0.0116	0.0135	0.0132	0.0128	0.0171	0.0143	0.0135

(c) Avg. Inference Time (s) vs Model Variation

Fig. 15.Model Variation Comparison Charts

Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

lopment for the up-scaled applications.

The SVM kernel we used could not perform multi-label classification. So, our approach was to train a single SVM kernel that takes a single region of a solar panel. As it requires expensive preprocessing time of processing patches to get the feature values and then running the model nine times, the inference time increased significantly, as we can see the results value for SVM (Segment)'s total average execution time in Table III. Even the single run time increased since we increased the kernel size for feature calculation to improve performance. Overall, we failed to find a consistent set of feature indices that yielded the best performance during this project. The best we achieved was 87.4% accuracy while testing.

We tried different classifiers, including fully connected layers (Multilayer Perceptron, MLP) and Recurrent Neural Networks (RNN) [23], to determine whether we could achieve better performance with hand-engineered features. In the case of MLP, we ended up with seven fully connected layers with ReLU [24] and batch normalization [25] for each layer, which significantly increased the size and inference time compared to the SVM (segment). Still, it failed to achieve an accuracy above 91%.

Next, we selected an RNN to improve accuracy on our one-dimensional feature series. A combination of CNNs and RNNs is a common approach to improve performance [26]. We applied a recurrent layer to the input feature vector, followed by five fully connected layers. We could achieve marginal success, maintaining similar performance while having a smaller memory footprint than MLP.

Unlike in binary classification, most of our success in 3x3 classification came from classifiers with automatically generated features. The models we found performed best include ResNet101, ResNet50, U-Net, and CNN1. The multilabel variants of these models differ in their output shapes, as shown in the model diagrams in the previous sections. The exception to this is the U-Net model, where we observed better performance after increasing its fully connected layer sizes.

TABLE III
FEATURE EXPLORATION MODEL RESULTS. THE SEGMENT SVM, MLP, AND RNN MODELS PERFORM NINE SEPARATE INFERENCES, ONCE FOR EACH SEGMENT, WHILE 3X3 MODELS RUN ONCE TO GET DIRT DETECTION RESULTS FOR THE NINE SUB-AREA.

Model	Accuracy (%)	Size (MB)	Inference Time (s)
ResNet101	100.00	162.74	2.78
ResNet50	100.00	89.99	1.65
U-Net	98.15	124.65	8.54
CNN1	98.15	86.67	0.380
CNN2	98.15	2.55	0.0317
CNN3	98.15	0.45	0.0280
CNN4	98.15	0.40	0.0266
CNN5	94.44	0.05	0.0247
SVM (Binary)	100.00	0.0045	0.0034
ResNet101 (3x3)	99.59	162.74	3.06
ResNet50 (3x3)	97.27	89.99	1.77
U-Net (3x3)	91.15	218.46	10.23
CNN1 (3x3)	92.18	86.67	0.408
MLP	91.0	3.08	0.66 (total: 5.9)
RNN	90.4	2.36	0.60 (total: 5.4)
SVM (Segment)	87.4	0.77	0.199 (total: 1.8)

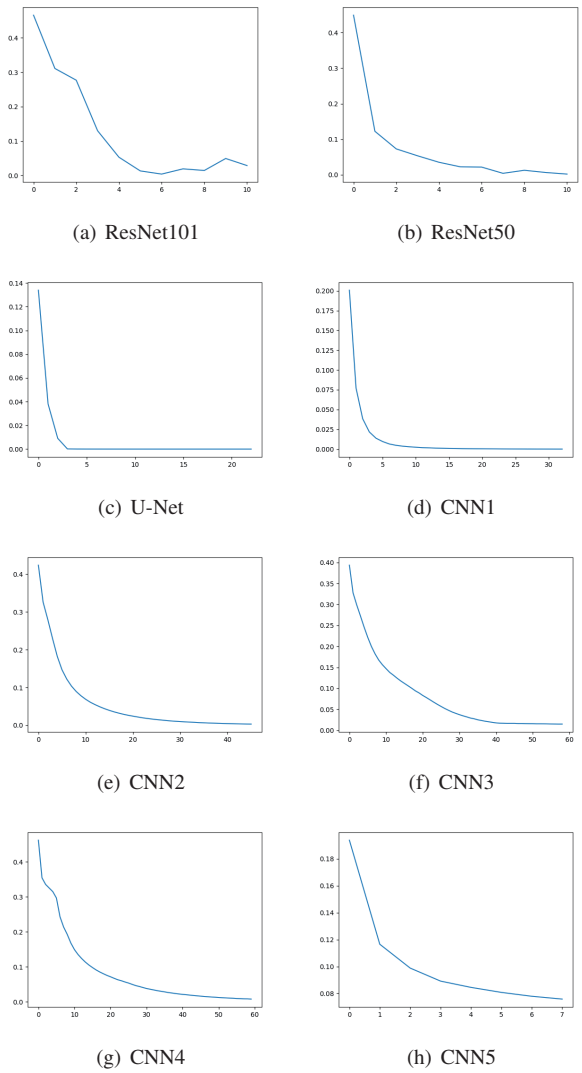
Unlike the segment SVM, feature MLP, and feature RNN, the 3x3 deep learning models do not need to split the image into segments and then perform binary classification on each one separately. Both ResNets and CNN1 outperformed the feature models in both accuracy and total inference time. The ResNet models in particular achieved higher accuracies with reasonable inference times compared to the other 3x3 classification models.

Fig. 17 and Fig. 18 show convergence via the loss versus epoch graphs for each of the 3x3 segment classification and feature classification deep learning models.

V. Conclusions and Future Works

Data-driven controller design is better suited to account for different types of dirt under various weather conditions. However, the hardware requirements for running data-driven algorithms tend to be higher; it is essential to optimize their performance to meet the target performance and power efficiency.

To develop a power-efficient algorithm using a data-driven approach, we first developed a dataset and hardware systems for algorithm development and testing. We used the dataset for the development and performance evaluation of selected algorithms. The definition of the dirt species in the dataset was used to set up performance comparison experiments for two different cleaning



(a) ResNet101 (b) ResNet50

(c) U-Net (d) CNN1

(e) CNN2 (f) CNN3

(g) CNN4 (h) CNN5

Fig. 16. Loss vs. Epoch Graphs for Binary Models

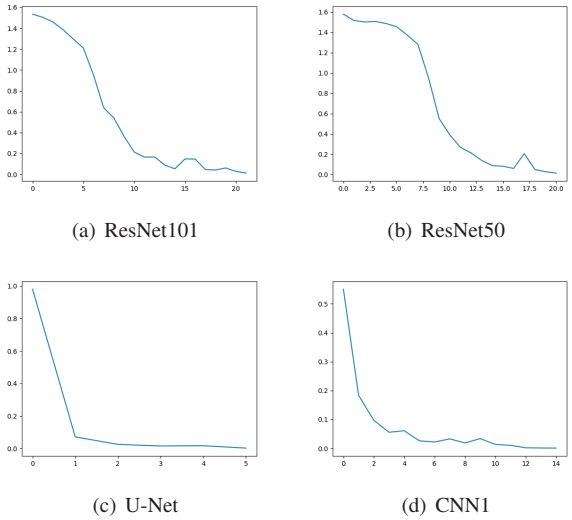
Ganghee Jang, Cameron Robinson, Towards Smart Solar Panel System Development: Case Study from Cleaning System Design and Algorithm Development Using Data-Driven Method, Journal of Energy, vol. 75 Number 2 (2026), 3–13
https://doi.org/10.37798/2026752743

mechanisms. We could assess that both systems were powerful enough to be used and select the best algorithms for our purpose among the chosen algorithms.

Then, we developed and compared algorithms from two perspectives: hand-engineered and automatically generated features. The algorithm using hand-engineered features outperformed in terms of power efficiency and inference time, while maintaining comparable performance for detecting dirt on the panel.

However, the development of the multi-zoned dirt detection took significant development time, and we failed to achieve the better performance than 91% of accuracy. On the contrary, the methods based on automatically generated features showed a relatively consistent trend across model size, inference size, and performance.

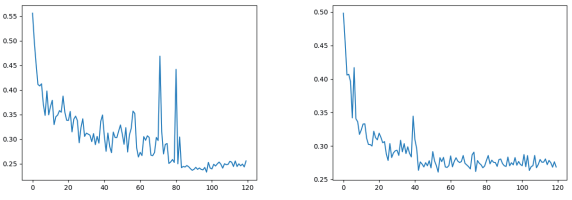
Admittedly, there are still many dirt species and surface conditions from other weather conditions in Oregon and other US areas. We need further study to determine whether the algorithms based on hand-engineered features remain useful as the dataset becomes more complex. We plan to update the dataset further, conduct the study, and make it publicly available.



(a) ResNet101 (b) ResNet50

(c) U-Net (d) CNN1

Fig. 17. Loss vs. Epoch Graphs for 3x3 Segment Models



(a) Feature MLP (b) Feature RNN

Fig. 18. Loss vs. Epoch Graphs for Feature Models

References

[1] J. F. Derakhshandeh, R. AlLuqman, S. Mohammad, H. AlHussain, G. AlHendi, D. AlEid, and Z. Ahmad, "A comprehensive review of automatic cleaning systems of solar panels," vol. 47. Publisher: Elsevier.

[2] R. J. Rupin, I. Hasan, R. Jahan, M. R. Haque Ontor, and M. M. Islam, "Dirt identification and removal based on non-contact infrared temperature sensor on solar panel," in *2023 7th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)*, pp. 971–977. [3] W. Hicks, "Scientists studying solar try solving a dusty problem," Apr. 2021. https://www.nrel.gov/news.

[4] B. O. Olorunfemi, N. I. Nwulu, and O. A. Ogbolumani, "Solar panel surface dirt detection and removal based on arduino color recognition," vol. 10, p. 101967.

[5] M. R. Habib, M. S. Tanvir, A. Y. Suhan, A. Vadher, S. Alam, T. T. Shawmee, K. Ahmed, and A. Alrashed, "Automatic solar panel cleaning system based on arduino for dust removal," in *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pp. 1555– 1558, 2021.

[6] M. K. Ghodki, "An infrared based dust mitigation system operated by the robotic arm for performance improvement of the solar panel," vol. 244, pp. 343–361.

[7] S. Maddali, "Clearing the dust: How CNNs and transfer learning can detect dust on solar panels."

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," vol. 86, no. 11, pp. 2278–2324.

[9] X. Hu, Z. Du, and F. Wang, "Research on detection method of photovoltaic cell surface dirt based on image processing technology," vol. 14, no. 1, p. 16842.

[10] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," vol. 13, no. 4, pp. 18–28.

[11] F. Wang, C. Zhang, S. Chen, G. Ying, and J. Lv, "Engineering hand-designed and deeply-learned features for person *Re*-identification," vol. 130, pp. 293–298.

[12] M. Budnik, E.-L. Gutierrez-Gomez, B. Safadi, D. Pellerin, and G. Quenot, "Learned features versus engineered features for multimedia´ indexing," vol. 76, no. 9, pp. 11941–11958.

[13] J. A. Bogovic, G. B. Huang, and V. Jain, "Learned versus hand-designed feature representations for 3d agglomeration."

[14] C.-W. Wang, J.-J. Ding, and L.-A. Chen, "Haze detection and haze degree estimation using dark channels and contrast histograms," in *2015 10th International Conference on Information, Communications and Signal Processing (ICICS)*, pp. 1–5.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), Lecture Notes in Computer Science, pp. 234–241, Springer International Publishing.

[16] U. Michelucci, "An introduction to autoencoders."

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition."

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, E. Tompson, N. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilimbi, and B. Prabhakaran, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8026– 8037, 2019.

[19] O. R. developers, "Onnx." https://onnx.ai/, 2021.

[20] M. Wang, "onnx/neural-compressor."

[21] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?," in *Proceedings of Machine Learning and Systems* (I. Dhillon, D. Papailiopoulos, and V. Sze, eds.), vol. 2, pp. 129–146.

[22] O. R. developers, "Quantize onnx models." https://onnxruntime. ai/docs/performance/model-optimizations/quantization.html# quantizing-an-onnx-model, 2025.

[23] R. M. Schmidt, "Recurrent neural networks (RNNs): A gentle introduction and overview."

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448– 456, PMLR.

[26] I. Sajedian, J. Kim, and J. Rho, "Finding the optical properties of plasmonic structures by image processing using a combination of convolutional neural networks and recurrent neural networks," vol. 5, no. 1, p. 27.