Applied Computer Science 2025, vol. 21, no. 4, pp. 32–46
https://doi.org/10.35784/acs_7589

Submitted: 2025-04-17 | Revised: 2025-05-18 | Accepted: 2025-06-01

Keywords: solar panel, dust detection, MobileNet, Sparse Shuffle layer

An CONG TRAN [1*], *Nghi CONG TRAN* [1]
[1*] Can Tho University, Can Tho, Vietnam, tcan@ctu.edu.vn, tcnghi@ctu.edu.vn
[*] Corresponding author: tcan@ctu.edu.vn

SSAtt-SolNet: An efficient model for dusty solar panel classification with Sparse Shuffle and Attention mechanisms

Abstract

This study introduces SSAtt-SolNet, a novel deep learning approach designed to detect dusty solar panels, thereby improving the efficiency and reliability of solar photovoltaic systems. The proposed model uses MobileNetV3 as its backbone to balance accuracy and computational efficiency. We introduced a novel sparse shuffle block that combines depth-separable convolution with a shuffle layer to improve model performance. We also incorporated an attention mechanism in the classification layer to selectively focus on relevant features while minimizing noise interference. This lightweight approach was evaluated on two public and one self-collected dataset containing a total of 10,118 images. The model was benchmarked against eight SOTA models in image classification and dusty solar panel detection using four metrics: accuracy, model parameters, model size, and floating point operations (FLOPs). The experimental results showed that our approach outperformed all baseline models, achieving the smallest standard deviation over five folds (99.68 ± 0.3%). Furthermore, the proposed model had the smallest size, the fewest parameters, and the minimum GFLOPs (0.1005). The paired t-test confirmed that the accuracy of our model is statistically significantly higher than all baseline models at the 95% confidence level. These results suggest that our proposed model is feasible for use in environments with limited computing resources.

1. INTRODUCTION

In recent years, environmental problems caused by the burning of fossil fuels have led many countries to focus on the development and use of renewable energy sources, especially solar energy, for electricity generation (Van Nguyen, 2023; Ogbolumani & Nwulu, 2022). This shift has led to a rapid increase in the use of solar photovoltaic (PV) systems as a sustainable source of electricity generation. However, the efficiency of these systems is significantly impacted by environmental factors, particularly the accumulation of dust on the surfaces of PV panels. Dust creates a barrier that reduces light transmission, resulting in significant energy loss. Research has shown that dust deposition can reduce the efficiency of PV panels by 20% or more, depending on various environmental conditions and the characteristics of the dust (Ahmed et al., 2013; Costa et al., 2016; Javed et al, 2017; Qasem et al., 2016; Shairi et al., 2020). This issue is particularly critical in arid and semi-arid regions, where frequent dust storms further contribute to the accumulation of particles on solar panels (Sarver et al., 2013). The decrease in power output reduces system efficiency and accelerates module degradation. In addition, unplanned maintenance to clean dirt from solar panels can result in significant financial losses (Alnasser et al., 2020). Therefore, it is critical to continuously monitor the cleanliness of solar panels to ensure optimal power generation (Shairi et al., 2020).

Traditional methods for managing dust buildup on photovoltaic (PV) panels typically involve periodic cleaning schedules that rely on manual inspections or physical sensors to estimate dust levels. However, these traditional approaches have several challenges, including high labor costs, time, and safety risks associated with accessing large installations or rooftop arrays (Dantas et al., 2020). While physical sensors can be helpful, they are often susceptible to environmental degradation and may not provide accurate, real-time data, making them unreliable for long-term monitoring (Javed et al., 2017). Given these limitations, there is an urgent need for an efficient, cost-effective, and non-invasive method to monitor dust accumulation on PV panels.

Recent advances in deep learning techniques have created new opportunities for automating complex detection tasks, particularly in image-based applications (LeCun et al., 2015). Convolutional neural networks (CNNs) and other deep learning models have proven effective in various visual inspection tasks, ranging from

detecting cracks in infrastructure to classifying soil types. These models have demonstrated the potential to accurately identify dust patterns on photovoltaic (PV) panels. A deep learning-based approach can analyze image data to detect dust without the need for physical interaction, enabling real-time and autonomous monitoring. This solution significantly reduces the reliance on manual inspections and sensor-based monitoring, while maintaining high levels of accuracy and scalability.

This research is motivated by the need to improve the efficiency and operational reliability of solar photovoltaic (PV) systems through an innovative deep learning approach to dust detection. Using convolutional neural networks (CNNs) and image recognition techniques, this study aims to develop a scalable, accurate, and autonomous dust detection solution. This will not only improve solar energy production, but also support the broader transition to renewable energy by optimizing PV system performance, reducing maintenance costs, and extending the lifetime of solar panels.

In this study, we use a MobileNetV3 backbone (Howard et al., 2019) to balance accuracy and computational efficiency in our dust detection model. MobileNetV3 is known for its lightweight architecture, making it particularly suitable for real-time applications in resource-constrained environments such as remote solar farms. We propose a novel sparse shuffle block that combines depth-separable convolution with a shuffle layer to further improve the efficiency of the model. This block reduces computational complexity by focusing on the most salient features, which improves the speed of the model without compromising accuracy.

The major contributions of this research are as follows:

1. We created a dedicated dataset for the detection of dust on solar panels by collecting images from various sources on the Internet. Each image was pre-processed and manually annotated to ensure high quality annotations. This makes the dataset a valuable resource for future research in this area.
2. We introduced a novel Sparse Shuffle Block in our model architecture to reduce the number of parameters and computational overhead. This block employs grouped operations along with channel shuffling to optimize computational efficiency, while depth-separable convolutions ensure that the model maintains high performance with fewer parameters.
3. We proposed a classification module that integrates an attention layer to further improve classification accuracy. This layer selectively emphasizes relevant features while suppressing irrelevant ones, thereby improving model focus and reliability.
4. We conducted extensive experiments to compare our approach with some state-of-the-art (SOTA) models on the same dataset. We evaluated the performance on several metrics, including accuracy, number of parameters, model size, and Giga Floating Point Operations (GFLOPs). Our results indicate that the proposed approach achieves a competitive balance between accuracy and efficiency, making it highly suitable for real-time, large-scale dust detection applications in solar PV maintenance.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces our proposed methodology. Section 4 outlines the experimental setup and training process. Section 5 discusses the experimental results. Finally, Section 6 summarizes the main results and concludes the paper and the future work.

2. RELATED WORK

Dust detection on photovoltaic (PV) solar panels has been an active area of research. Methods have evolved from traditional sensor-based and image processing techniques to more advanced AI and deep learning approaches. Traditional approaches, such as thermal imaging and IoT-based sensor systems, have provided the foundation for automated dust detection, albeit with limitations in real-time capabilities and resource efficiency. Recent advances in machine learning (ML) and deep learning (DL) have introduced novel solutions that improve accuracy, scalability, and feasibility of deployment in diverse environments. This section reviews these methods, starting with traditional approaches and moving to machine learning and deep learning techniques that address the unique challenges of real-time, large-scale dust detection on PV systems.

2.1. Traditional dust detection approaches for solar panels

Several methods have been used to detect dust on solar panels, including thermal imaging, IoT with sensor-based techniques, and digital imaging. Thermal imaging uses infrared cameras to capture the heat emitted by panels, effectively detecting performance changes and defects. However, it requires high-quality equipment and sophisticated software, which can be cost prohibitive (Chaudhary & Chaturvedi, 2017; Cubukcu &

Akanalci, 2020). In contrast, IoT with sensor-based approaches facilitates both detection and periodic cleaning, providing a cost-effective solution, although it often suffers from sensor degradation over time (Mohammed et al., 2018; Zainuddin et al., 2019).

Recently, digital image processing techniques have been introduced to capture and analyze images of PV panels, using linear regression and spectral decomposition methods to detect dust with up to 90% accuracy. However, the effectiveness of these approaches can vary depending on the type of dust and environmental conditions (Abuqaaud & Ferrah, 2020; Tribak & Zaz, 2019). Each of these methods has unique advantages and limitations, highlighting the need for continuous monitoring and cost-effective maintenance strategies to ensure optimal energy output from solar panels.

2.2. Machine learning approaches in dust detection

Recent advances in Artificial Intelligence (AI)-based approaches have opened new perspectives in dust detection and have gained significant traction. Various machine learning (ML) techniques have been applied in various domains to improve detection accuracy and reliability. For example, Igathinathane et al. (2009) applied machine vision techniques to analyze airborne dust particles, while Maitre et al. (2019) used a random forest algorithm for mineral classification, achieving 90% accuracy. These studies highlight the robust applications of ML in image analysis and classification, showing how ML techniques improve dust detection and enable effective categorization of particle properties.

In another approach, Proietti et al. (2015) used the k-nearest neighbors (kNN) algorithm to investigate dust accumulation rates, analyzing the nearest data points in multidimensional space to accurately characterize dust particles. These studies illustrate the versatility of ML in various dust detection contexts. However, they reveal limitations when scaling to large PV arrays, as they may lack the real-time capabilities required for continuous monitoring and maintenance in high-resolution, large-scale applications.

2.3 Deep learning for dust detection on solar panels

In recent years, deep learning has made significant progress in fields such as computer vision, and has shown great potential for automated dust detection on solar panels. State-of-the-art object detection algorithms such as YOLO (You Only Look Once) (Guo et al., 2022) and RetinaNet (Wang et al., 2019) are capable of delineating the location and boundaries of solar panels in images with high accuracy. However, these models typically require significant computational resources, making them difficult to deploy in real-time, resource-constrained environments. Image segmentation techniques such as U-Net (Siddique et al., 2021) and Mask R-CNN (Bharati & Pramanik, 2020) have also been used to accurately identify the location and condition of PV panels. While effective, these methods generate numerous parameters and require considerable computational power, complicating the maintenance process.

Research exploring the use of CNN models specifically for dust detection has yielded promising results. For example, Maity et al. (2020) applied the LeNet CNN with custom dropout and pooling layers and achieved 80% accuracy in binary classification of dust presence, while Zyout & Oatawneh (2020) experimented with deep CNNs and found that AlexNet achieved 93.3% accuracy on a labeled dataset of solar panel images. Other studies have investigated architectures such as ResNet (He et al., 2016), EfficientNet (Koonce, 2021), and MobileNet (Howard et al., 2019; Sinha & El-Sharkawy, 2019), each with a focus on computational efficiency and accuracy. While these architectures have proven effective, challenges remain due to the high demand for hyperparameter tuning and the need for extensive training data, which can hinder training efficiency and model deployment in real-time environments (LeCun et al., 2015; Shao et al., 2023).

2.4. Optimizing deep learning models for real-time deployment

Given the constraints of deploying deep learning models on resource-constrained devices, there has been a notable shift toward optimizing neural network architectures for real-time, energy-efficient applications. MobileNet and ShuffleNet are examples of lightweight CNN architectures designed for this purpose, balancing accuracy with reduced computational requirements. Advanced architectures, such as MobileNetV3 (Howard et al., 2019), integrate attention mechanisms and architectural optimizations to improve performance without significantly increasing model size. In addition, transfer learning is increasingly used to apply pre-trained models to the dust detection task, leveraging existing model weights to achieve high accuracy with minimal retraining.

Data augmentation techniques, such as random cropping, flipping, and color jittering, are also critical for improving model robustness, allowing deep learning models to better generalize across different environmental and dust conditions (Shorten & Khoshgoftaar, 2019). Studies have shown that such augmentation strategies are particularly beneficial in applications such as solar panel dust detection, where data collection is challenging and variations in lighting and dust accumulation must be accounted for.

In summary, recent advances in AI and deep learning have led to innovative methods for solar panel dust detection. However, challenges remain in balancing accuracy and computational efficiency, especially for real-time applications in resource-constrained environments. This study addresses these issues by proposing an optimized deep learning framework that integrates MobileNetV3 with split-shuffle block and attention mechanisms. This combination aims to provide a highly accurate and scalable solution for dust detection. Our approach is designed for real-time deployment, making it suitable for large-scale photovoltaic (PV) system maintenance and dust monitoring applications.

3. METHODOLOGY

This section describes the architecture and design principles of SSAtt-SolNet (Sparse Shuffle Attended MobileNet), a binary classification model designed to determine whether images of solar panels are clean or dirty. This study proposes a deep learning-based approach to efficiently detect dirty solar panels, addressing a critical challenge in photovoltaic system maintenance and performance optimization. Our model architecture has three core components to balance computational efficiency and high classification accuracy. First, we employ a feature extraction module based on MobileNetV3 that captures essential visual features while maintaining lightweight computation. Second, we introduce a novel sparse shuffle channel block that optimizes the use of convolutional resources to improve feature discrimination between dusty and clean solar panels. Finally, we propose a classifier block augmented with an attention mechanism that further refines the feature representation to enable accurate classification. This methodology leverages state-of-the-art design principles to provide a robust and scalable solution for dusty solar panel detection. The architectural design of the model is described in Figure 1. The algorithm for building the SSAtt-SolNet model described above is shown in Algorithm 1 (See Appendix).

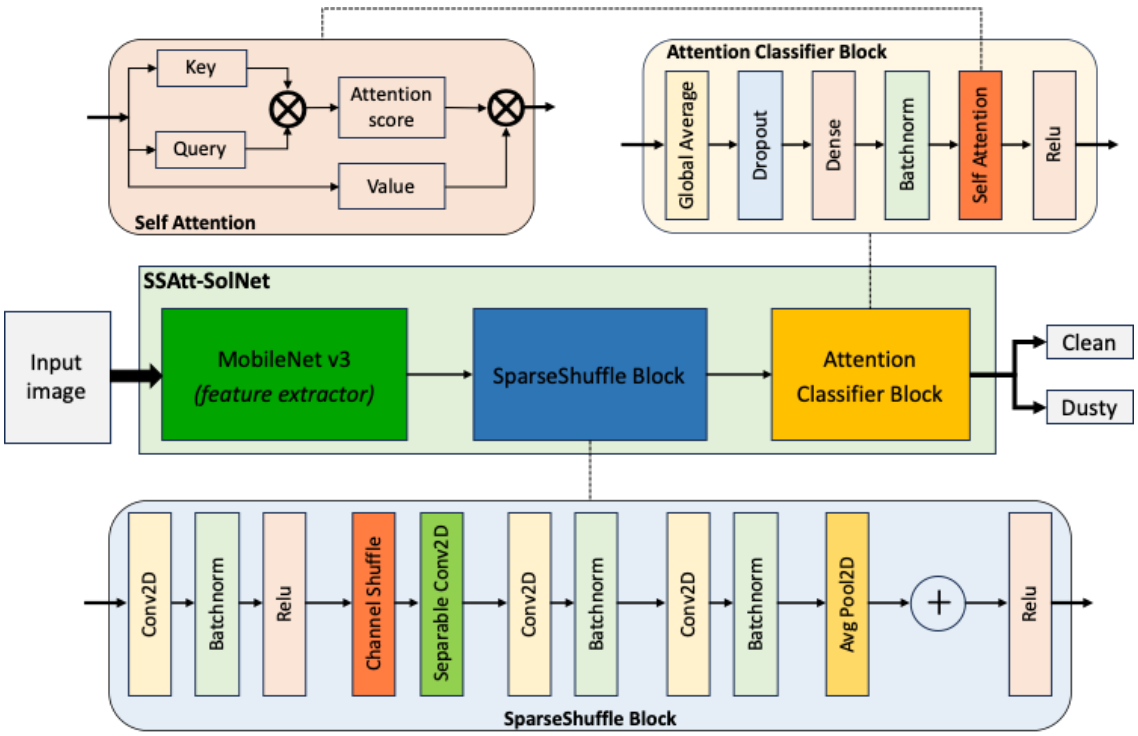

Fig. 1. SSAtt-SolNet architecture

3.1. Feature extraction with MobileNetV3

In the feature extraction block, we use MobileNetV3, a state-of-the-art lightweight convolutional neural network (CNN), which is characterized by its remarkable efficiency and outstanding performance in various vision-related tasks. We used the pre-trained MobileNetV3 model on the ImageNet dataset, which provides a strong foundation for extracting meaningful features. Using this pre-trained model can significantly reduce both the training time and computational cost associated with our specific task. Our model uses only the layers that precede the Adaptive Average Pooling class in MobileNetV3, as shown in Figure 2. These layers are specifically designed to capture spatial and hierarchical features from input images while minimizing resource consumption. By selectively using these layers, our model focuses only on relevant feature extraction, while the subsequent blocks in our architecture handle the classification task.

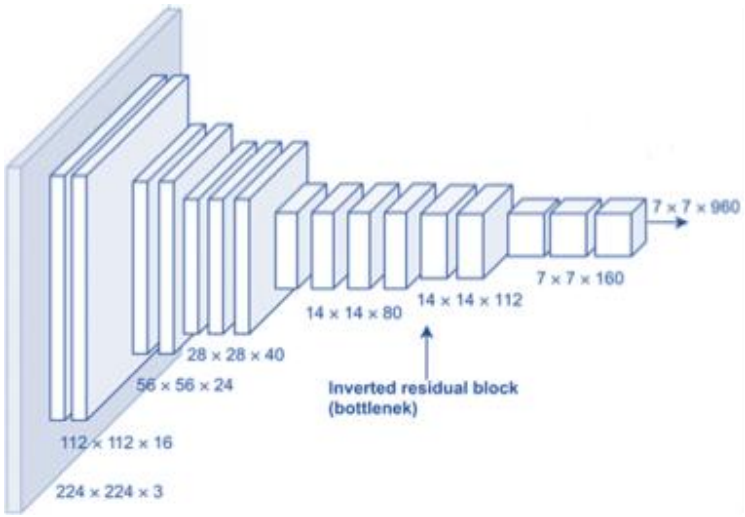

Fig. 2. MobileNetV3 as the model's backbone for feature extraction

The model was also fine-tuned on our custom dataset to adapt the feature extractor to the domain of dusty and clean solar panels. Fine-tuning allows the pre-trained model to adjust its learned weights to better capture features specific to solar panels, such as texture, lighting variations, and dust accumulation patterns. This process effectively improves the model's ability to distinguish between clean and dusty panels. This stage of the architecture balances computational efficiency with accurate feature representation, providing a strong foundation for subsequent processing and classification.

3.2. Sparse Shuffle Block

The Sparse Shuffle Block is a novel architectural component designed to improve the efficiency and performance of convolutional neural networks (CNNs) for classifying clean and dusty solar panels. This block aims to achieve a balance between computational efficiency and robust feature representation by combining depth-separable convolution with channel shuffling. This makes it particularly suitable for resource-constrained scenarios such as edge devices and real-time solar panel monitoring systems. The main components of this block are described below.

3.2.1 Depthwise Separable Convolutions

The Sparse Shuffle block uses the Depthwise Separable Convolutions architecture to reduce computational cost while maintaining spatial filtering capabilities. This approach splits a standard convolution operation into two stages:

1. Depthwise Convolution: Depthwise convolution convolves each feature channel independently, focusing on spatial filtering within each channel. This approach significantly reduces the number of parameters and computations compared to traditional convolutions, making it an efficient alternative for lightweight and resource-constrained deep learning applications. Figure 3 illustrates a deep

convolution, where $D_{in} \times D_{in}$ is the size of the input image, $D_{Out} \times D_{OUT}$ is the size of the output image, M is the number of channels, and $D_k \times D_k \times 1$ is the kernel size.

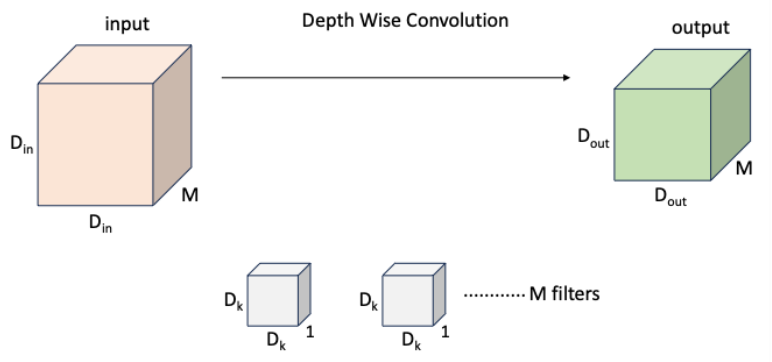

Fig. 3. Depthwise Convolution operation

For an input tensor $X \in \mathrm{R}^{D_{in} \times D_{in} \times M}$ a filter tensor $F \in \mathrm{R}^{D_k \times D_k \times 1}$ and an output tensor $Y \in \mathrm{R}^{D_{out} \times D_{out} \times M}$. For each channel $m \in \{1, 2, ..., M\}$ and spatial location (i, j) the output is calculated by equation 1 as follows:

$$Y(\mathrm{i}, \mathrm{j}, \mathrm{m}) = \sum_{m_1=0}^{D-1} \sum_{m_2=0}^{D_k-1} F(m_1, m_2, m) \cdot \mathrm{X}(\mathrm{s} \cdot \mathrm{i} + \mathrm{m}_1, \mathrm{s} \cdot \mathrm{i} + \mathrm{m}_2, \mathrm{m}) \quad (1)$$

Where: $D_{out} = \left[\frac{D_{in} + 2p - D_k}{s} \right] + 1$,

s – strike,

p – padding.

2. Pointwise Convolution (1×1): Pointwise convolution is a lightweight operation that combines and integrates information across feature channels. By applying a 1×1 convolution kernel, this step refines the feature representations and establishes critical inter-channel correlations. Figure 4 illustrates the pointwise convolution, where $D_{in} \times D_{in}$ is the size of the input image, M is the number of channels in the input image, N is the number of channels in the output image, and 1×1×M is the kernel size. This design ensures efficient channel-wise interaction without increasing spatial complexity.

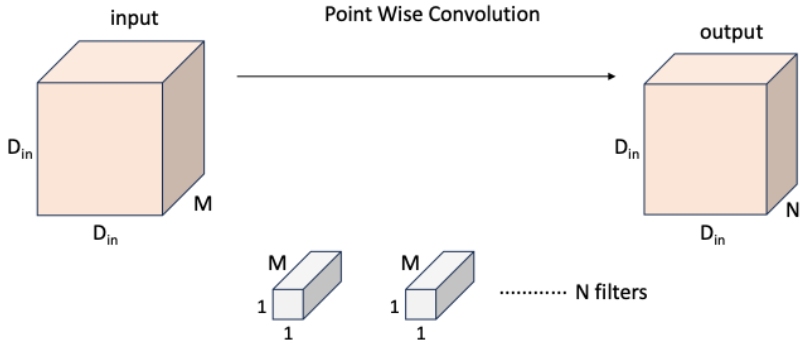

Fig. 4. Pointwise Convolution operation

For an input tensor $X \in \mathrm{R}^{D_{in} \times D_{in} \times M}$, a pointwise filter tensor $F \in \mathrm{R}^{1 \times 1 \times M \times N}$ and an output tensor $Y \in \mathrm{R}^{D_{in} \times D_{in} \times N}$. For each spatial location (i, j) and each output channel n, the output is calculated as in Equation 2.

$$Y(i, j, n) = \sum_{m=0}^{M} F(1, 1, m, n) \cdot X(i, j, m) \quad (2)$$

The block learns compact and efficient feature representations by decomposing the standard convolution into these two operations, while significantly reducing computational complexity.

3.2.2. Channels shuffling

Following the Depthwise Separable Convolutions, the Channel Shuffling mechanism was employed to ensure effective mixing of feature channels. This mechanism is inspired by the ShuffleNet architecture to overcome the limitations of grouped convolutions, where channels within certain groups are processed independently.

Let C be the input channel, and the channel shuffling process can be shown in Figure 5 and described as follows:

- Step 1: Divide C channels of input data into g groups, each group containing n channels.
- Step 2: Transform the grouped data from [N, H, W, g×n] to [N, H, W, g, n].
- Step 3: Transform the data in step 2 from [N, H, W, g, n] to [H, H, W, n, g].
- Step 4: Transform the data in step 3 from [N, H, W, n, g] to [N, H, W, n×g].

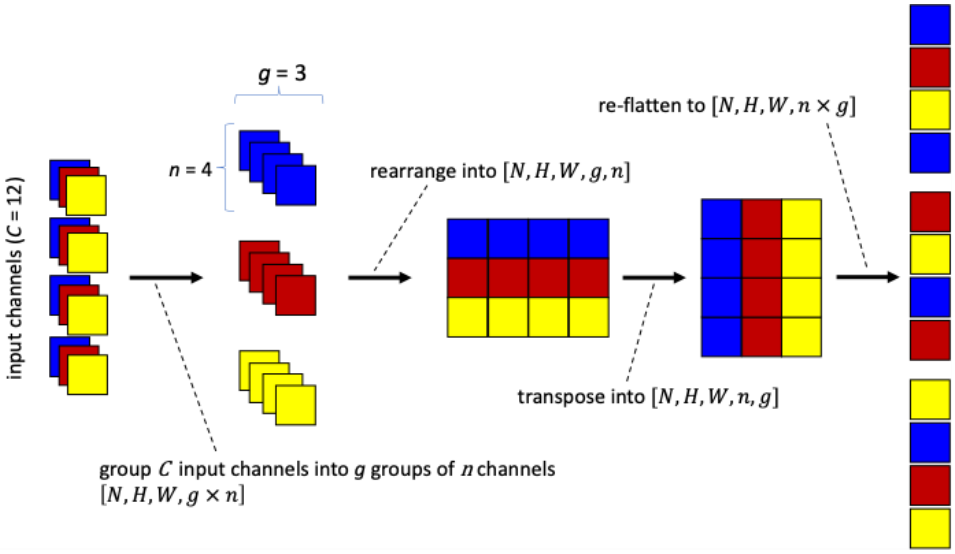

Fig. 5. Channel Shuffling mechanism

By mixing feature channels, this mechanism provides the following benefits:

1. Reorganization across groups: Channel shuffling reorganizes channels to facilitate data sharing among different groups, effectively eliminating the independence commonly associated with grouped convolutions.
2. Enhanced information sharing: This process enhances communication among feature maps, boosting the diversity and robustness of the learned representations.

By combining Depthwise Separable Convolution, Channel Shuffling operations, Pointwise Convolution, and an optional residual link, the SparseShuffle block provides an efficient and effective approach to feature processing. Depthwise Separable Convolutions separate spatial filtering from channel shuffling, allowing these distinct operations to significantly reduce the number of parameters and computational cost compared to traditional convolutions. Channel Shuffling further enhances the effectiveness of the block by promoting information exchange between channels, ensuring that features are effectively combined and redistributed across channels. Finally, Pointwise Convolution helps to restore connectivity between channels, while the residual connection improves training stability and promotes gradient flow.

Together, these techniques enable the proposed SparseShuffle block to achieve a balance between computational efficiency and feature representation power, improving the model's ability to accurately distinguish between dusty and clean solar panels. In addition, its lightweight design makes it particularly well-suited for scalable deployment in resource-constrained environments, such as edge computing systems used for real-time solar panel monitoring.

3.3. Attention classifier block

The final component of our proposed model is the classification block. This block acts as the final stage in the pipeline, where the features extracted from the previous layers are refined and categorized for accurate classification. It consists of two main components: an attention layer and a dense classifier pipeline. The attention layer helps the model focus on the most relevant features, improving feature representation and classification performance. Meanwhile, the dense classifier pipeline transforms these features into the final categories: "dusty" or "clean". Details on this block are described below.

3.3.1. Attention layer

The attention layer is the core mechanism that emphasizes the most relevant features within the feature map. The attention mechanism enhances critical information while reducing noise by exploiting the relationships between different spatial and channel components. Its process includes:

1. Key, query, and value representation: The feature embeddings are divided into key, query, and value vectors. These vectors model the interactions between different parts of the feature space.
2. Attention score computation: Attention scores are derived from the dot product of the query and key vectors, followed by a normalization step (e.g., softmax) that quantifies the relevance of each feature.
3. Feature refinement: The normalized attention scores are applied to the value vector, improving feature maps by focusing on the most informative elements. This step improves the model's ability to capture important patterns and contextual information.

This mechanism ensures optimal exploitation of spatial and interchannel relationships, significantly increasing the discriminative power of the feature representation.

3.3.2. Dense classifier pipeline

Following the attention layer in the classification module is a systematic pipeline for feature aggregation and final prediction. This pipeline combines spatial and channel-wise information processing, feature learning, and regularization techniques to improve classification performance. The components of the pipeline are:

1. Global Average Pooling (GAP): The GAP layer reduces the spatial dimensions of feature maps while preserving important channel-wise feature statistics. This is achieved by averaging each feature map over its spatial dimensions. This operation condenses global spatial information while preserving the integrity of channel-specific features. This layer can then focus on the most significant global patterns.
2. Dropout regularization: The dropout layer is incorporated to reduce the risk of overfitting by randomly setting a portion of the input units to zero during training. This stochastic regularization technique allows the development of robust feature representations by preventing dependence on specific neurons.
3. Dense (Fully Connected) Layers: The fully connected layers are used to transform the refined feature embeddings from the previous layers into the desired output space. These layers provide the model with powerful learning capabilities, allowing it to capture complex, non-linear feature interactions that are essential for accurate classification.
4. Batch Normalization: Batch normalization is applied after the dense layers to improve the stability of the training process and speed up convergence. This technique normalizes feature activations across the entire batch to a mean of zero and unit variance. This helps reduce internal covariate shifts during training and improves model robustness.

4. EXPERIMENTAL SETUP AND TRAINING

In this section, we present the experimental setup and training procedures used to evaluate the performance of the proposed model. This includes the description of the experimental dataset, setup, and training configuration. We also describe the evaluation metrics used in our analysis.

4.1. Evaluation dataset

We used two publicly available datasets and a manually curated dataset to evaluate the proposed model. The curated dataset was designed to improve the model's performance in real-world applications by including

images taken under different conditions and with different objects and backgrounds. Dataset information is described in Table 1.

Tab. 1. Experimental datasets

	Dataset 1	Dataset 2	Dataset 3	Total
Clean	1,300	1,492	2,910	5,332
Dusty	1,101	1,069	2,616	4,786
All	2,231	2,561	5,326	10,118

4.1.1. Datasets 1 and 2: Public dataset

The first experimental dataset is a public dataset of solar photovoltaic panel images taken in Bangladesh with minimal extraneous content (Onim et al., 2022). Most of the images in this dataset clearly focus on solar panels without significant background clutter, making it easy to classify their conditions. The dataset contains a total of 2,231 images, with 1,300 clean panels and 1,101 dusty panels.

The second public dataset was used by Alatwi et al. (2024). This dataset also consists of images of clean and dusty solar panels taken from different vantage points, at different times of day, and with different types of dust. The total number of images in this dataset is 2,561, with the number of clean panels being 1,492 and the number of dusty panels being 1,069.

4.1.2. Dataset 3: Self-developed dataset

To improve the generalization and performance of the model in real-world scenarios, we developed an additional dataset from different online images. This dataset is challenging because it contains, among other things, solar panels, different backgrounds, and different environmental conditions. It contains 1,343 images, of which 817 are labeled clean and 526 are labeled dirty. The inclusion of different scenes, camera angles, and lighting creates a realistic test environment for detecting dust on solar panels. The combination with Dataset 1 ensures that the model is trained under different conditions, increasing the reliability of our results.

4.2. Experimental setup

To improve the robustness and effectiveness of our training process, and in particular to address the problem of overfitting, we have implemented an early stopping mechanism. This mechanism continuously monitors the validation loss during training. If there is no observable improvement in the validation loss after a given number of epochs, the training process is automatically stopped.

We have extensively investigated different hyperparameter values and their results to optimize the training process. Based on our findings, we have established the following detailed training configuration.

- Optimizer: The Adam optimizer was chosen for its efficient learning properties and its ability to accelerate convergence. A learning rate of 0.001 was selected to balance the learning speed with the stability of the model.
- Loss function: Binary cross-entropy was used as the loss function, which is well suited for the binary classification task of dust detection.
- Batch size: A batch size of 32 was chosen to optimize the trade-off between memory usage and training stability.
- Epochs: The model was trained for a maximum of 50 epochs, with early stopping triggered if the validation accuracy did not improve over 10 consecutive epochs to prevent overfitting and ensure efficient training.

At the end of the 5-fold cross-validation, we calculated the mean accuracy and the standard deviation across all folds. This served as the primary metric for evaluating and comparing the performance of our proposed model against the baseline models. This approach ensured a thorough and consistent evaluation of the model's effectiveness in detecting dust on solar panels, thereby enhancing its relevance for real-world applications.

To thoroughly evaluate the performance of our proposed model, we compared it with several well-established baseline models that are commonly used in image classification tasks. Specifically, our model was benchmarked against ResNet50, VGG16, InceptionV3, EfficientNetB0, EfficientNetB4, MobileNetV2, and MobileNetV3Small. These architectures are well-known for their strong performance and efficiency in visual

recognition tasks. Furthermore, we also included Solnet (Onim et al., 2022), a recent model specifically designed for dust detection on solar panels, as a baseline for comparison.

Each baseline model was initialized with pre-trained weights from the ImageNet dataset to leverage transfer learning. To ensure a fair comparison, we standardized the training process using consistent hyperparameters, including the same learning rate, batch size, and data loader configuration. This approach allows any performance differences to be attributed solely to the architectures, minimizing potential biases from experimental conditions.

4.3 Evaluation metrics

We evaluated our proposed system using three main metrics: accuracy, number of model parameters, and GFLOPs (Giga Floating Point Operations Per Second). Together, these metrics thoroughly assess the model's predictive ability, complexity, and computational efficiency, providing insight into the practical applicability of the system in real-world scenarios.

Accuracy is defined by Equation 3 as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{3}$$

Where: TP (True Positive) – the number of dirty solar panels correctly identified as dirty,
TN (True Negative) – the number of clean solar panels correctly identified as clean,
FP (False Positive) – the number of clean panels mistakenly identified as dirty,
FN (False Negative) – the number of dirty panels mistakenly identified as clean.

The number of model parameters reflects the complexity of the model by indicating the total number of trainable weights, which affects memory usage and the risk of overfitting. GFLOPS (Giga Floating Point Operations per Second) measures computational demand by quantifying how many billion operations the model performs per second. This metric is important for evaluating computational cost and scalability, especially in resource-constrained environments.

5. EXPERIMENTAL RESULTS

Table 2 shows the detailed comparison of our model, SSAtt-SolNet, with the baseline models, focusing on accuracy over five cross-validation folds, the number of parameters, the model size, and the computational complexity (GFLOPs).

The experimental results show that our proposed method, SSAtt-SolNet, outperforms the state-of-the-art baseline models across various performance metrics, including accuracy, computational efficiency, and model complexity. The evaluation is based on five-fold cross-validation to ensure robustness and generalizability.

The accuracy results indicate that SSAtt-SolNet achieves good performance, achieving an average accuracy of 99.68% ± 0.3% across all five folds. The standard deviation of 0.3% demonstrates the model's consistency and stability, as it maintains uniform accuracy without significant fluctuations. In contrast, the baseline models, such as InceptionV3 (96.63% ± 1.28%), EfficientNetB4 (95.6% ± 1.03%), and ResNet50 (95.24% ± 1.23%), show lower average accuracies and a greater degree of variability across folds. Paired t-tests confirm that the differences in accuracy between SSAtt-SolNet and these models are statistically significant at a confidence of 95%, further validating the reliability of SSAtt-SolNet performance.

In particular, the comparison with VGG16 (93.76 ± 11.94)% shows an interesting result. Although VGG16 produces a lower overall accuracy than SSAtt-SolNet, the paired t-test shows that the difference is not statistically significant ($p = 0.229$). This can be attributed to the relative stability of VGG16's performance in several folds (e.g., folds 2, 3, and 4), where its accuracy is comparable to SSAtt-SolNet. However, VGG16 experienced a sharp drop in accuracy in Fold 1 (84.42%), which significantly lowers its average accuracy and increases its standard deviation (±11.94). This large variation across folds indicates that VGG16's performance is less consistent and more sensitive to variation than SSAtt-SolNet in the datasets used.

In contrast, models such as ResNet50 and EfficientNetB0 have consistently lower accuracy across multiple folds than the SSAtt-SolNet results. For example, ResNet50 records lower results in fold 3 (94.39%) and fold 5 (93.49%), while EfficientNetB0 achieves similar performance in folds 1 and 5, but fails to achieve the same

level of stability as SSAtt-SolNet overall. These patterns explain the statistically significant differences compared to SSAtt-SolNet.

Tab. 2. Experimental results

Model	Accuracy (%)						No of parameters	Model size (MB)	GFLOPs
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average			
Resnet50	95.96	96.18	94.39	96.18	93.49	95.24 ± 1.23	24,702,593	94.23	7,732
VGG16	84.42	98.65	99.15	97.55	98.61	95.68 ± 6.32	15,043,137	57.39	30,714
InceptionV3	94.84	97.75	96.41	96.18	97.98	96.63 ± 1.28	22,917,665	87.42	5,687
EfficientNetB0	91.92	95.29	96.18	96.18	94.61	94.84 ± 1.76	4,771,236	18.2	0.7886
EfficientNetB4	96.18	96.41	96.41	94.84	94.18	95.6 ± 1.03	18,657,632	71.17	3,052
MobileNetV2	91.7	94.17	96.41	94.61	93.27	94.03 ± 1.73	2,979,469	11.37	0.601
MobileNet V3-Small	90.58	93.04	90.35	90.35	92.6	91.38 ± 1.32	1,300,337	4.96	0.1138
Solnet	89.01	86.99	86.77	89.46	87.66	87.98 ± 1.2	46,756,609	178.36	2,013
Our Method (SSAtt-SolNet)	100.00	100.00	99.32	99.55	100.00	99.68 ± 0.3	672.674	2.57	0.1005

The variability observed among the baseline models, particularly VGG16, highlighted the importance of stability across different folds as a measure of robustness. SSAtt-SolNet not only achieved the highest overall accuracy, but also demonstrated low variability. This level of stability strengthens the reliability of SSAtt-SolNet for practical applications, as the model can effectively generalize to different subsets of the data without experiencing the fold-specific fluctuations seen in baseline methods.

In addition to accuracy, SSAtt-SolNet also achieves remarkable efficiency in terms of model size, number of parameters, and computational cost. With only 672,674 parameters, it occupies only 2.57 MB of memory, making it significantly lighter than other models such as ResNet50, which has 24.7 million parameters and requires 94.23 MB, and EfficientNetB4 with 18.6 million parameters and 71.17 MB. This simplification in model structure is achieved without compromising performance, underscoring the effective optimization in the design of SSAtt-SolNet for this application. In addition, the computational cost, measured in GFLOPs, further highlights its efficiency. SSAtt-SolNet requires only 0.1005 GFLOPs, a fraction of the computational requirements of models such as VGG16 (30.714 GFLOPs) and ResNet50 (7.732 GFLOPs). These results indicate that SSAtt-SolNet is more accurate and practical in resource-constrained environments such as mobile and embedded systems.

The baseline models illustrate different trade-offs between accuracy, computational cost, and model size. For example, InceptionV3 achieves a competitive accuracy of 96.63 ± 1.28, but requires significantly more computational resources (5,687 GFLOPs) and a larger model size (87.42 MB). Similarly, EfficientNetB4 offers strong accuracy (95.6 ± 1.03) with reduced computational cost compared to InceptionV3, but still falls short of SSAtt-SolNet's performance. On the other hand, lightweight models such as MobileNetV3-Small (91.38 ± 1.32) and MobileNetV2 (94.03 ± 1.73) are computationally efficient but exhibit significantly lower accuracy, reflecting the trade-off between performance and efficiency in these models.

Overall, the experimental results clearly show that SSAtt-SolNet significantly outperforms the baseline models in terms of both accuracy and consistency. The paired t-test analysis provides strong statistical evidence for this conclusion, as four of the five baseline models had statistically significantly lower performance than SSAtt-SolNet.

6. CONCLUSIONS AND FUTURE WORK

In this work, we proposed a dusty solar panel detection model based on sparse shuffle and attention mechanisms. Inspired by the idea of ShuffleNet, we combined depth-separable convolution with channel shuffling to form the sparse shuffle mechanism. This combination maximizes model performance while maintaining low computational requirements. Depthwise Separable Convolution decomposes a standard

convolution operation into two distinct stages, allowing the model to learn more compact and efficient feature representations. Then, the sparse channel shuffling mechanism was used to improve the interaction between feature channels. This mechanism ensures that channels that are initially processed independently in groups are mixed together to promote better information exchange.

In addition, we implemented an attention mechanism in the classification layer to improve the model's ability to focus on the most relevant features. This improved the relationships between channels and the use of spatial context, resulting in better feature extraction and classification accuracy.

The proposed model was evaluated and compared with several baseline models for image classification and dusty solar panel detection on three datasets. The evaluation results show that our proposed model, SSAtt-SolNet, achieves a remarkable balance between accuracy, efficiency, and robustness. It consistently outperforms almost all baseline models in terms of accuracy while maintaining a lightweight architecture and low computational cost. Statistical analysis confirmed that the accuracy improvements are significant compared to most competitive models. These results suggest that our model is not only an optimal solution for achieving state-of-the-art accuracy, but also highly practical for use in computationally constrained environments and real-time applications.

The Sparse Shuffle Block introduced in this work has potential for a wide range of applications that require efficient and lightweight convolutional neural networks, beyond the detection of dusty solar panels. Its design effectively balances computational efficiency with strong feature representation, making it suitable for real-time image processing tasks on resource-constrained edge devices such as mobile phones, drones, and embedded vision systems. In addition, this block can be applied in various fields, including industrial quality control, autonomous vehicles, portable medical imaging devices, and augmented reality (AR) or virtual reality (VR) systems, where both accuracy and efficiency are critical. This versatility highlights the value of the Sparse Shuffle Block as a fundamental architectural building block for deep learning models intended for real-time, low-power deployment scenarios.

In our future work, we plan to investigate more sparse shuffling techniques to improve efficiency and adaptability to varying levels of dust accumulation. In addition, we will explore more attention mechanisms, including multi-scale or self-supervised, to better capture spatial and contextual features. We will also explore alternative lightweight architectures for feature extraction, such as EfficientNet, ShuffleNet, and architectures based on Neural Architecture Search (NAS), to evaluate their feature extraction performance. Finally, we will consider additional research directions, such as the use of random data augmentation techniques, the application of quantization methods, etc.

Conflicts of Interest

The authors declare no conflict of interest.

REFERENCES

Abuqaaud, K. A., & Ferrah, A. (2020). A novel technique for detecting and monitoring dust and soil on solar photovoltaic panel. *2020 Advances in Science and Engineering Technology International Conferences (ASET)* (pp. 1–6). IEEE. https://doi.org/10.1109/ASET48392.2020.9118377

Ahmed, Z., Kazem, H.A., & Sopian, K. (2013). Effect of dust on photovoltaic performance: Review and research status. *Latest trends in renewable energy and environmental informatics*, *34*(6), 193–199. https://doi.org/10.13140/2.1.4527.9523

Alatwi, A. M., Albalawi, H., Wadood, A., Anwar, H., & El-Hageen, H. M. (2024). Deep learning- based dust detection on solar panels: A low-cost sustainable solution for increased solar power generation. *Sustainability*, *16*(19), 8664. https://doi.org/10.3390/su16198664

Alnasser, T. M., Mahdy, A. M., Abass, K. I., Chaichan, M. T., & Kazem, H. A. (2020). Impact of dust ingredient on photovoltaic performance: An experimental study. *Solar Energy*, *195*, 651–659. https://doi.org/10.1016/j.solener.2019.12.008

Bharati, P., & Pramanik, A. (2020). Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. In A. K. Das, J. Nayak, B. Naik, S. K. Pati, & D. Pelusi (Eds), *Computational Intelligence in Pattern Recognition* (Vol. 999, pp. 657–668). Springer Singapore. https://doi.org/10.1007/978-981-13-9042-5_56

Chaudhary, A. S., & Chaturvedi, D. K. (2017). Thermal image analysis and segmentation to study temperature effects of cement and bird deposition on surface of solar panels. *International Journal of Image, Graphics and Signal Processing*, *9*(12), 12-22. https://doi.org/10.5815/ijigsp.2017.12.02

Costa, S. C., Diniz, A. S. A., & Kazmerski, L. L. (2016). Dust and soiling issues and impacts relating to solar energy systems: Literature review update for 2012–2015. *Renewable and Sustainable Energy Reviews*, *63*, 33–61. https://doi.org/10.1016/j.rser.2016.04.059

Cubukcu, M., & Akanalci, A. (2020). Real-time inspection and determination methods of faults on photovoltaic power systems by thermal imaging in Turkey. *Renewable Energy*, *147*, 1231–1238. https://doi.org/10.1016/j.renene.2019.09.075

Dantas, G. M., Mendes, O. L. C., Maia, S. M., Alexandria, A. R. (2020). Dust detection in solar panel using image processing techniques: A review. *Research, Society and Development*, *9*(8), 321985107–321985107. https://doi.org/10.33448/rsd-v9i8.5107

Guo, Z., Wang, C., Yang, G., Huang, Z., & Li, G. (2022). MSFT-YOLO: Improved YOLOv5 based on transformer for detecting defects of steel surface. *Sensors*, *22*(9), 3467. https://doi.org/10.3390/s22093467

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.-C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., & Le, Q. (2019). Searching for MobileNetV3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 1314–1324). IEEE. https://doi.org/10.1109/ICCV.2019.00140

Igathinathane, C., Melin, S., Sokhansanj, S., Bi, X., Lim, C. J., Pordesimo, L. O., Columbus, E. P. (2009). Machine vision based particle size and size distribution determination of airborne dust particles of wood and bark pellets. *Powder Technology*, *196*(2), 202–212. https://doi.org/10.1016/j.powtec.2009.07.024

Javed, W., Guo, B., & Figgis, B. (2017). Modeling of photovoltaic soiling loss as a function of environmental variables. *Solar Energy*, *157*, 397–407. https://doi.org/10.1016/j.solener.2017.08.046

Koonce, B. (2021). *Convolutional neural networks with Swift for Tensorflow: image recognition and dataset categorization*. Apress.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Maitre, J., Bouchard, K., B´edard, L. P. (2019). Mineral grains recognition using computer vision and machine learning. *Computers & Geosciences*, *130*, pp. 84–93. https://doi.org/10.1016/j.cageo.2019.05.009

Maity, R., Shamaun Alam, Md., & Pati, A. (2020). An approach for detection of dust on solar panels using CNN from RGB dust Image to predict power loss. In P. K. Mallick, P. K. Pattnaik, A. R. Panda, & V. E. Balas (Eds), *Cognitive Computing in Human Cognition* (Vol. 17, pp. 41–48). Springer International Publishing. https://doi.org/10.1007/978-3-030-48118-6_4

Mohammed, H. A., Baha'a, A., & Al-Mejibli, I. S. (2018). Smart system for dust detecting and removing from solar cells. *Journal of Physics*, *1032*(012055). 10.1088/1742-6596/1032/1/012055

Ogbolumani, O., & Nwulu, N. (2022): A food-energy-water nexus meta-model for food and energy security. *Sustain Prod Consum*, *30*, 438–453. https://doi.org/10.1016/j.spc.2021.12.019

Onim, M. S. H., Sakif, Z. M. M., Ahnaf, A., Kabir, A., Azad, A. K., Oo, A. M. T., Afreen, R., Hridy, S. T., Hossain, M., Jabid, T., & Jabid, T. (2022). Solnet: a convolutional neural network for detecting dust on solar panels. *Energies*, *16*(1), 155. https://doi.org/10.3390/en16010155

Proietti, A., Panella, M., Leccese, F., & Svezia, E. (2015). Dust detection and analysis in museum environment based on pattern recognition. *Measurement*, *66*, 62–72. https://doi.org/10.1016/j.measurement.2015.01.019

Qasem, H., Mnatsakanyan, A., & Banda, P. (2016). Assessing dust on PV modules using image processing techniques. *2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC)* (pp. 2066–2070). IEEE. https://doi.org/10.1109/PVSC.2016.7749993

Sarver, T., Al-Qaraghuli, A., Kazmerski, L. L. (2013). A comprehensive review of the impact of dust on the use of solar energy: History, investigations, results, literature, and mitigation approaches. *Renewable and Sustainable Energy Reviews*, *22*, 698–733. https://doi.org/10.1016/j.rser.2012.12.011

Sayyah, A., Horenstein, M. N., & Mazumder, M. K. (2014). Energy yield loss caused by dust deposition on photovoltaic panels. *Solar Energy*, *107*, 576–604. https://doi.org/10.1016/j.solener.2014.05.030

Shairi, N. A. S., Ghoni, R., & Ali, K. (2020). Solar panel dust monitoring system. Eng. Herit. J 4(2), 44–45. https://doi.org/10.26480/gwk.02.2020.44.45

Shao, Y., Fan, S., Sun, H., Tan, Z., Cai, Y., Zhang, C., & Zhang, L. (2023). Multi-scale lightweight neural network for steel surface defect detection. *Coatings*, *13*(7), 1202. https://doi.org/10.3390/coatings13071202

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Siddique, N., Paheding, S., Elkin, C. P., & Devabhaktuni, V. (2021). U-net and its variants for medical image segmentation: A review of theory and applications. *IEEE Access*, *9*, 82031–82057. https://doi.org/10.1109/ACCESS.2021.3086020

Sinha, D., & El-Sharkawy, M. (2019). Thin mobilenet: An enhanced mobilenet architecture. *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)* (pp. 0280–0285). IEEE. https://doi.org/10.1109/UEMCON47517.2019.8993089

Tribak, H., & Zaz, Y. (2019). Dust soiling concentration measurement on solar panels based on image entropy. *2019 7th International Renewable and Sustainable Energy Conference (IRSEC)* (pp. 1–4). IEEE. https://doi.org/10.1109/IRSEC48032.2019.9078250

Van Nguyen, T. (2023). Applications of artificial intelligence in renewable energy: A brief review. *2023 International Conference on System Science and Engineering (ICSSE)* (pp. 348–351). IEEE. https://doi.org/10.1109/ICSSE58568.2023.10227160

Wang, Y., Wang, C., Zhang, H., Dong, Y., & Wei, S. (2019). Automatic ship detection based on RetinaNet using multi-resolution Gaofen-3 imagery. *Remote Sensing*, *11*(5), 531. https://doi.org/10.3390/rs11050531

Zainuddin, N. F., Mohammed, M., Al-Zubaidi, S., & Khogali, S. I. (2019). Design and development of smart self-cleaning solar panel system. *2019 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)* (pp. 40–43). IEEE. https://doi.org/10.1109/I2CACIS.2019.8825071

Zyout, I., & Oatawneh, A. (2020). Detection of PV solar panel surface defects using transfer learning of the deep convolutional neural networks. *2020 Advances in Science and Engineering Technology International Conferences (ASET)* (pp. 1–4). IEEE. https://doi.org/10.1109/ASET48392.2020.9118279

APPENDIX

SSAtt-Solnet model training algorithm.

Algorithm 1: SSAtt-SolNet Algorithm

Input : *imageShape*: Shape of the input image (*width*, *height*, *channels*).
numClasses: Number of output classes.
Output: Compiled SSAtt-SolNet model.

```
/* Define a new input layer */
inputLayer = CreateInputLayer(shape = imageShape, name='input');
/* Resize images to fixed dimensions (224, 224) */
resizedInput = Resizing(inputLayer, (224, 224));
/* Preprocess input layer using preprocessor of MobileNet V3 */
preprocessedInput = MobileNetV3Preprocessor(resizedInput);
/* Define a backbone using a pre-trained MobileNetV3Small model */
backbone = MobileNetV3Small(input_shape=imageShape,
  include_top=False, weights='imagenet');
/* Identify the layer of the backbone for extracting features (layer 29th) */
baseModel = CreateModel(backbone.input, backbone.layers[-29].output);
/* Extract features using the base model of the backbone */
features = baseModel.extract(preprocessedInput);
/* Perform the shuffle separable operation (see Algorithm 2) */
shuffledFeatures = ShuffleSeparable(features);
/* Global average pooling */
glbAvgPooling = GlobalAveragePooling2D(shuffledFeatures);
/* Apply dropout for regularization */
dropedOut = Dropout(glbAvgPooling, 0.5);
/* Define pre-classification fully connected layers including a Dense layer with ReLU
   activation followed by a Batch normalization operator */
preClassificationLayer = CreateSequential([
  Dense(32, activation='relu'),
  BatchNormalization()]);
/* Apply pre-classification layers */
preClassified = preClassificationLayer(dropedOut);
/* Add self-attention mechanism */
selfAtt = SelfAttention([preClassified, preClassified]);
/* Define the classification head (output layer) */
predictionLayer = CreateDenseLayer(numClass, activation='sigmoid');
/* Output classification results */
outputs = predictionLayer(selfAtt);
return outputs;
```

Algorithm 2: Shuffle Separable Algorithm – ShuffleSeparable()

Input : x: Input tensor.
noOfGroups: number of groups for grouped convolutions.
noOfChannels: number of output channels.
strideSize: Stride size for spatial down-sampling.
Output: Processed tensor after applying shuffle separable convolutions.

begin
  /* Save the original input tensor for residual connections */
  $orgInput = x$;
  /* Apply 1×1 grouped convolution */
  $convInput$ = Conv2D(x, $noOfChannels$ // 4, $kernelSize$=1, $strides$=(1, 1), $padding$='same', $groups$=$noOfGroups$;
  $batNorInput$ = BatchNormalization($convInput$) ; // Normalize feature maps
  $reluInput$ = ReLU($batNorInput$) ; // Apply ReLU activation
  /* Shuffle channels (see Algorithm 3) */
  $scInput$ = ChannelShuffling($reluInput$, $noOfGroups$);
  /* Apply depthwise separable convolution */
  $sepConvInput$ = SeparableConv2D($scInput$, $noOfChannels$ // 4, $kernelSize$=3, $strides$=$strideSize$, $padding$='same');
  $sepConvInput2nd$ = Conv2D($sepConvInput$, $noOfChannels$ // 4, $kernelSize$=1, $strides$=1, $padding$='valid');
  $batNorInput2nd$ = BatchNormalization($sepConvInput2nd$);
  /* Add residual connections or concatenate down-sampled outputs */
  if $strideSize == (1, 1)$ **then**
    /* Element-wise addition for residual connection */
    $batNorInput2nd$ = Add([$batNorInput2nd$, $originalInput$]);
  else if $strideSize == (2, 2)$ **then**
    /* Down-sample original input */
    $agvPool2dInput$ = AvgPool2D($orgInput$, (3, 3), $strides$=(2, 2), $padding$='same');
    /* Concatenate with down-sampled input */
    $concatDownSampled$ = Concatenate([$batNorInput2nd$, $agvPool2dInput$]);
  /* Apply ReLU activation */
  reLuInput = ReLU(concatDownSampled);
  return reluInput // Return the processed tensor
end

Algorithm 3: Channel Shuffling Algorithm – ChannelShuffling()

Input : x – A tensor that contains the data to be channel-shuffled
Output: Channel-shuffled tensor

begin
  /* Get the input tensor properties */
  $width$, $height$, $channels$ = x.shape;
  /* //: integer (floor) division operator */
  $channelGroups$ = x.numberOfchannels // $noOfGroups$;
  /* Reshape the tensor to split channels into groups */
  $resizedInput$ = Resharp(x, $width$, $height$, $channelGroups$, $noOfGroups$;
  /* Permute dimensions to shuffle channels within groups */
  $permutedInput$ = Permute($resizedTensor$, [1, 2, 4, 3]);
  /* Reshape back to the original channel structure */
  $shuffledTensor$ = Reshape($permutedInput$, [$width$, $height$, $channels$]);
  return $shuffledTensor$;
end