SolarGuard: Intelligent Defect Detection on Solar Panels using Deep Learning

Abstract

The SolarGuard project addresses the critical need for automated, high-efficiency inspection of Photovoltaic (PV) solar panels. This project utilized Transfer Learning with the MobileNetV2 architecture to rapidly train a robust image classification model capable of detecting six distinct panel defects. Through aggressive data augmentation and disciplined training protocols, the model achieved a **Validation Accuracy of approximately** 75.00%. The final performance on unseen data demonstrated a consistent result, yielding a **Test Accuracy of** 70.37%. This result highlights specific challenges in distinguishing between closely related defect types like 'Dusty' and 'Bird-drop'. The resulting system is deployed as a user-friendly Streamlit web application, providing instant defect classification and actionable maintenance insights.

1. Introduction

1.1 Problem Statement

Solar energy infrastructure relies on the continuous optimal performance of individual PV panels. Defects such as physical damage, soiling (dust, bird droppings), and electrical faults drastically reduce power output. Early and accurate identification of these faults is essential for maximizing the return on investment (ROI) of solar farms.

1.2 Project Goal

The primary objective of SolarGuard was to develop and deploy an intelligent defect classification system capable of analyzing panel images instantly to categorize faults with high accuracy, thereby streamlining maintenance operations.

2. Methodology

2.1 Data Acquisition and Preparation

The dataset consisted of images categorized into six classes, representing common solar panel conditions:

1. Clean
2. Dusty
3. Bird-drop

4. Electrical-damage
5. Physical-Damage
6. Snow-Covered

To enhance generalization capabilities and prevent overfitting on the limited dataset, extensive **Data Augmentation** was applied, including: rotation (10 degrees), width and height shifting (0.05), and zooming (0.05). The dataset was split into Training (80%), Validation (10%), and Test (10%) subsets, with stratified sampling to preserve class distribution.

2.2 Model Architecture: Transfer Learning

Transfer Learning was utilized to leverage the powerful feature extraction capabilities of MobileNetV2, a computationally efficient pre-trained Convolutional Neural Network (CNN).

- **Base Model:** MobileNetV2 pre-trained on the ImageNet dataset. The base layers were initially **frozen** (base_model.trainable=False) to retain learned features and ensure rapid convergence during the initial training epochs.
- **Custom Classification Head:** A sequence of layers was added on top of the MobileNetV2 feature maps:
    - Flatten
    - Dense(512,activation='relu')
    - Dropout(0.5) for regularization
    - Dense(6,activation='softmax') for final classification

2.3 Training Configuration

Parameter	Value	Purpose
Optimizer	Adam	Standard efficient optimization algorithm.
Loss Function	Categorical Crossentropy	Used for multi-class classification with one-hot encoded labels.
Batch Size	32	Standard mini-batch size.
Callbacks	EarlyStopping (patience=5)	Prevent overfitting and unnecessary computation.

Model Saving	ModelCheckpoint(save_weights_only=True)	Saved only the weights of the model achieving the lowest validation loss for robust deployment.
Stability Fix	Fixed Random Seed (SEED_VALUE=42)	Implemented across NumPy and TensorFlow to ensure reproducible results.

3. Results and Discussion

The model trained for 19 epochs, with the EarlyStopping callback intervening and restoring weights from Epoch 14.

3.1 Performance Metrics

The performance difference between validation and test sets is noted below:

Metric	Best Validation Result	Final Test Result	Epoch/Status
Loss	0.70703	0.9216	14 / Final Evaluation
Accuracy	75.00%	70.37%	14 / Final Evaluation

3.2 Analysis of Generalization and Class Performance

The 4.63% drop in accuracy (from 75.00% validation to 70.37% test) indicates mild overfitting. While significantly better than the previous run's large gap, the model's overall accuracy still suggests room for improvement in feature learning.

Detailed analysis of the classification report reveals key challenges:

- **Strong Performance:** The model excelled at identifying **Physical-Damage** (100% precision) and **Snow-Covered** panels (100% precision), indicating these features are distinct and non-ambiguous.
- **Weak Performance:** The model struggled most with distinguishing contamination: **Bird-drop** had the lowest recall (53%), suggesting many bird-drop images were

misclassified as other contaminants (e.g., 'Dusty'). **Dusty** also showed balanced but mediocre performance (63% F1-score).

- **Precision vs. Recall Trade-off:** The model generally favors precision over recall (the macro average for precision is 0.79, while recall is 0.71). This means that when the model identifies a defect, it is usually correct, but it fails to detect a significant number of actual defects.

3.3 Deployment Solution

The model was deployed using a Streamlit framework. A key technical challenge in deployment—loading the MobileNetV2 structure from the saved file—was circumvented by implementing a **model reconstruction strategy**. The `app.py` script explicitly rebuilds the MobileNetV2 base and the custom top layers, and then loads the saved weights file (best_classification_weights.weights.h5) into the correct structure, ensuring reliable operation of the web application.

4. Conclusion and Future Work

SolarGuard successfully utilizes deep learning and transfer learning principles to create an effective and deployable solution for solar panel defect detection, achieving a final **Test Accuracy of** 70.37%. The system provides actionable insights but requires further tuning to improve generalization, particularly for subtle soiling defects.

4.1 Future Enhancements (Addressing Performance and Ambiguity)

1. **Fine-Tuning:** Unfreeze the top few layers of the MobileNetV2 base and continue training with a very low learning rate to optimize the feature extractor specifically for solar panel images, focusing on distinguishing subtle contaminants.
2. **Increased Regularization:** Increase the Dropout rate (e.g., to 0.6 or 0.7) and/or introduce L2 kernel regularization to the Dense layers to further minimize the chance of overfitting.
3. **Dataset Expansion:** Acquire a larger, more varied dataset, particularly focusing on ambiguous or multi-defect images to improve generalization and strengthen the model's ability to classify 'Bird-drop' and 'Dusty' panels accurately.