European Journal of Technique

journal homepage: https://dergipark.org.tr/en/pub/ejt

Vol.14, No.2, 2024

Research Article

Automated Fault Classification in Solar Panels Using Transfer Learning with EfficientNet and ResNet Models

Rojbin Akinca[1], Hüseyin Firat[2] and Mehmet Emin Asker[3,*]

[1] Dicle University, Department of Renewable Energy Resources, Diyarbakır, Türkiye. (e-mail: rojbin.akinca@dicle.edu.tr).
[2] Dicle University, Department of Computer Engineering, Diyarbakır, Türkiye. (e-mail: huseyin.firat@dicle.edu.tr).
[3,*] Dicle University, Department of Electricity and Energy, Diyarbakır, Türkiye. (e-mail: measker@dicle.edu.tr).

ARTICLE INFO

Received: Aug., 15. 2024
Revised: Nov., 05. 2024
Accepted: Nov., 09. 2024

Keywords:
Solar panel
Transfer learning
EfficientNet
ResNet
Fault classification

Corresponding author: *Mehmet Emin Asker*

ISSN: *2536-5010* | e-ISSN: *2536-5134*

DOI: https://doi.org/10.36222/ejt.153783

ABSTRACT

Classifying and detecting faults in solar panels using deep learning methods is crucial to ensuring their efficiency and longevity. In this study, we propose a model that concatenates ResNet and EfficientNet to classify faults in solar panel images. ResNet's advantage lies in its residual connections, which help mitigate the vanishing gradient problem and improve training of deep networks. EfficientNet is known for its scalability and efficiency, providing a balanced trade-off between accuracy and computational cost by optimizing network depth, width, and resolution. Together, these models enhance fault classification accuracy while maintaining efficiency. To evaluate the performance of the proposed model, experimental studies were conducted using a solar panel dataset with six classes: bird-drops, covered snow, dusty, clean, electrical and physical damage on the surfaces of solar panels. The results demonstrated that the ResNet101 + EfficientNetB1 concatenation achieved superior performance, with an accuracy of 87.55%, F1-score of 88.13%, recall of 88.75%, and precision of 87.92%. This concatenation provided significant improvements in fault classification metrics compared to individual models.

1. INTRODUCTION

Recently, the reduction in fossil fuel reserves and the impact of global warming have driven a greater focus on sustainable, clean, and renewable energy (RE) sources such as geothermal energy, hydroelectric, solar, and wind [1,2]. Among RE sources, solar energy is one of the most powerful and is being used increasingly often [3]. Solar panels (SPs) are key to producing solar energy efficiently [4]. The performance and durability of SPs are closely connected to identifying and fixing potential issues. Therefore, early detection of faults in SPs is crucial for ensuring uninterrupted energy production and reducing maintenance costs [5-8].

The SP faults result from a combination of electrical irregularities and environmental factors. Various issues, such as pyhsical damage, electrical damage, hot spots, bird droppings, dust, and panel deformation due to external conditions, can cause disruptions in energy production [9-12]. Traditional machine learning methods for detecting these faults tend to be both costly and time-consuming. However, fault detection methods based on manual inspections and remote monitoring tools have been significantly improved recently with the advent of deep learning (DL) models [1,6]. The DL involves artificial intelligence techniques capable of recognizing and learning complex patterns from large datasets [13]. Convolutional neural networks (CNNs) and DL-based feature extractors are particularly effective in extracting precise information from image data and identifying faults [11,14]. Using DL methods to detect faults in SPs provides several advantages. Early fault detection can extend the lifespan of panels, reduce energy production losses, and make maintenance processes more efficient, thereby increasing the reliability of solar energy systems [15,17].

In the literature, there are various studies on fault detection from SP images using DL models. Some of these studies are as follows. Espinosa et al. [5] introduced an automated technique for classifying faults in SP images by employing CNNs for both classification and semantic segmentation based on RGB images. The method achieved an average accuracy 70% for categorizing four classes: dust, shadows, cracks, no-fault, and 75% for distinguishing between no-fault and fault categories. Le et al. [6] introduced a CNN framework that combines a residual network method with ensemble method to identify faults in photovoltaic modules. Following the method, they applied various transformations to augment the dataset, aiming to achieve the highest classification performance. Additionally, they determined the optimal number of filters by evaluating both the augmented datasets and raw with different filter configurations. The ensemble method achieved an accuracy of 94.40% for the 2-class problem and 85.90% for the 12-class problem. Duranay [11] examines using infrared images of solar modules for detecting defects via DL, aiming to improve solar energy system efficiency. The dataset included 20,000 images across 12 classes, with classification performed using an EfficientNetB0 model and SVM classifier. The method achieved strong results, with average accuracy of 93.93%, F1-score of 89.82%, precision of 91.50%, and sensitivity of 88.28%. Mahmud et al. [12] conducted fault detection on SP images using the VGG16 and VGG19 models. Their dataset, besides clean panels, included five additional classes: Snow-Covered, Physical damage, Electrical damage, Bird-drops, and Dusty. Their experimental results showed a prediction accuracy of 72.88% with VGG16 and 86.44% with VGG19. Alves et al. [18] introduced a CNN architecture for classifying faults in photovoltaic modules. They created a balanced dataset by applying oversampling and undersampling techniques. To analyze the classification effectiveness of their proposed approach, they examined different scenarios. The overall classification criterias for the 2-class problem were reported as 92.50% for accuracy (Acc), 92.00% for F1-score (F1s), recall (R), and precision (P). Additionally, They noted that the proposed approach acquired an Acc of 66.43% for the 11-class problem. Lee et al. [19] introduces LIRNet, a lightweight inception residual convolutional network designed for detecting faults in SPs. LIRNet leverages DL and hierarchical learning, consisting of two phases: data preprocessing using K-means clustering to refine the dataset, followed by model training to enhance fault detection accuracy and processing speed.

In this study, we propose concatenating ResNet and EfficientNet models to classify faults in SP images. ResNet is known for its ability to effectively train deep neural networks using residual connections, which help prevent issues like vanishing gradients. EfficientNet, on the other hand, optimizes accuracy and efficiency by scaling depth, width, and resolution systematically. By concatenating these models, the proposed approach benefits from ResNet's depth and learning capabilities and EfficientNet's balanced performance, resulting in improved fault classification accuracy. To test the effectiveness of the proposed model, a dataset containing images of SPs with six classes was used. Experimental results on this dataset showed that among the ResNet models, ResNet152 acquired the best performance with an Acc of 84.15%, P of 85.72%, R of 82.84%, and F1s of 83.97%. For the EfficientNet models, the best results were obtained with EfficientNetB2, achieving an Acc of 82.64%, P of 83.96%, R of 83.54%, and F1s of 83.67%, and with EfficientNetB4, achieving an Acc of 83.40%, P of 83.68%, R of 83.45%, and F1s of 83.41%. The concatenation of different versions of the two models resulted in the best performance with the ResNet101 + EfficientNetB1 model, achieving an Acc of 87.55%, P of 87.92%, R of 88.75%, and F1s of 88.13%. This concatenation showed improvements in classification metrics compared to the closest models by 3.4% in Acc, 2.2% in P, 5.21% in R, and 4.16% in F1s.

In the other sections of the paper, Section 2 covers the dataset containing solar panel images, the proposed model, and the models that make up the proposed model. Section 3 discusses the experimental setup, evaluation metrics, and experimental results in detail. The final section, Section 4, provides a general summary of the study's findings.

2. MATERIALS AND METHODS

2.1. Solar Panel (SP) Dataset

The presence of bird drops, snow, dust etc. on SP surfaces decreases their efficiency and the energy they generate. Thus, it is essential to monitor and clean these panels regularly. Creating an effective process for monitoring and cleaning SPs is crucial to enhance their efficiency, lower maintenance costs, and minimize resource usage.

In this study, a dataset containing publicly available SP images from the Kaggle platform was used [20]. This dataset consists of images of clean, dusty, and various damaged panels, categorized into six different classes. The classes are as follows: snow-covered, physical damage, electrical damage, bird drops, dusty, and clean. The original dataset includes a total of 885 SP images. Among these images, 193 are clean, 190 are dusty, 103 have electrical damage, 69 have physical damage, 123 are snow-covered, and 207 contain bird drops. The sample panel images in the dataset are shown in Figure 1.

The aim of utilizing this dataset is to evaluate the effectiveness of various machine learning classifiers in accurately classifying bird drops, snow, dust, as well as electrical and physical damage on the surfaces of SPs.

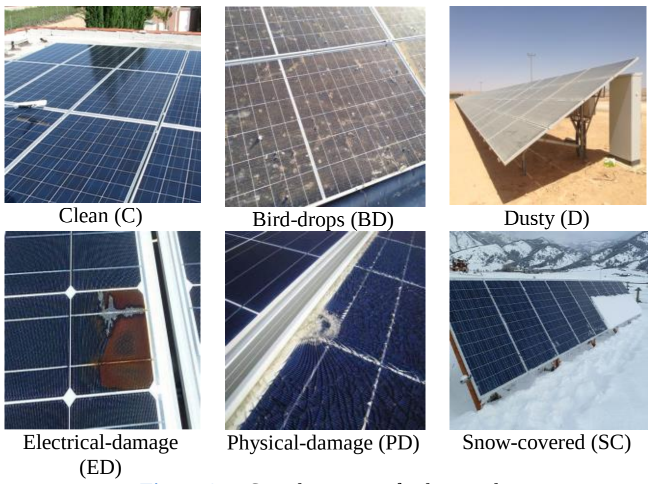


Figure 1. Sample images of solar panels

2.2. Transfer Learning

Transfer learning (TL) is an effective method in DL that involves utilizing a model trained for one specific task as the foundation for a model aimed at a different but related task [21]. This approach is particularly beneficial in DL, as it allows models to leverage features learned from large datasets, reducing the amount of data and training time required for new tasks [22]. The primary concept involves transferring knowledge from one domain (source) to another (target), where the source domain contains a large amount of data, while the target domain has only a small amount of data available.

The TL involves utilizing feature extraction from a model that has already been pre-trained, thereby avoiding the necessity for developers to train a model from scratch [21]. Typically, a TL model is trained on a large dataset such as ImageNet [23]. The learned parameters from this model can then be applied to a CNN-based model for a different but related application. Such models can be directly employed for making predictions on new tasks or integrated into the training processes of related applications [21,22].

In summary, the TL is a versatile and efficient approach to developing deep learning models, allowing the reuse of existing knowledge to solve new problems. It accelerates the training process, improves performance, and is particularly useful when working with limited data. By leveraging pre-trained models and applying techniques like feature extraction and fine-tuning, transfer learning can effectively address a wide range of tasks across different domains.

In this study, EfficientNet [24] and Residual Network (ResNet) [25], two important pre-trained models available in the Keras library, are used for classifying faults in solar panels.

2.3. Residual Network (ResNet) Model

Residual networks, or ResNets, are a type of deep CNN architecture introduced to address the challenges associated with training very deep networks. The key innovation of ResNets is the concept of residual learning (RL), which helps overcome issues like vanishing gradients and difficulty in training deep models [25].

The RL is a technique in deep learning designed to address the difficulties associated with training very deep CNN. Instead of learning the entire transformation directly from input to output, the RL focuses on learning the residual, or the difference between the desired output and the input [25].

In practice, this means that if the aim of the network is to learn a mapping $H(x)$, it learns $F(x)$=$H(x)$−x instead. Here, $H(x)$ is the desired mapping, x is input, and $F(x)$ represents the residual function. The final output of the network is then obtained by adding this residual to the original input, resulting in $F(x)$+x (Figure 2). This method simplifies the learning process by allowing the network to concentrate on the residuals, which are often easier to model than the complete transformation [25].

The RL is implemented through residual blocks, which are the core components of ResNets. Each block consists of several convolutional layers followed by a skip connection that adds the input of the block directly to its output (Figure 2). These shortcut or skip connections, ensure that the learning focuses on the residuals and help maintain the flow of gradients during training, thus mitigating the vanishing gradient problem. In addition, the benefits of the RL are substantial. It makes the training of very deep networks more manageable by simplifying the optimization process. By enabling the effective training of deeper models, the RL enhances feature extraction capabilities and overall model accuracy [25].

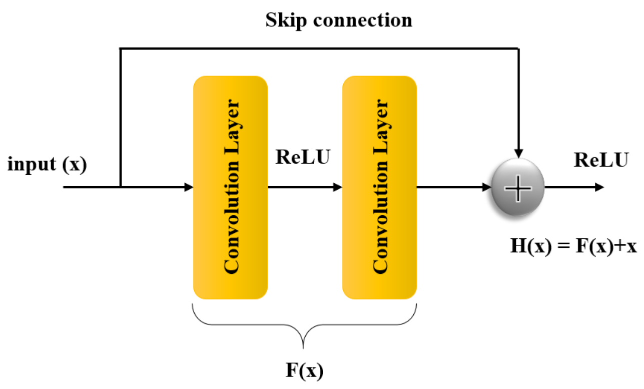


Figure 2. Block of residual learning

ResNets can be quite deep, with common variants including ResNet50, ResNet101, and ResNet152 where the number indicates the number of layers. This depth enables the network to learn more complex features and achieve high accuracy in various tasks [25]. ResNet50 model has 50 layers and is known for its balance between performance and computational efficiency. It is commonly used for various image recognition tasks. Suitable for applications where computational resources are limited, and real-time performance is needed. With 101 layers, ResNet101 model provides increased depth compared to ResNet50, allowing it to capture more complex features and achieve higher accuracy on image classification tasks. Ideal for tasks requiring more detailed feature extraction and where additional computational resources are available. It can be a good choice for medium to large-scale image classification tasks. The deepest among the three, with 152 layers, ResNet152 offers even greater accuracy, making it suitable for more demanding image recognition tasks. However, it requires more computational resources and longer training times.Best for high-performance applications that require the utmost accuracy, such as medical imaging and large-scale visual recognition [25].

2.4. EfficientNet Model

EfficientNet is a family of CNNs designed for image classification tasks, introduced by Google researchers. The key innovation in EfficientNet is its use of a compound scaling approach that uniformly scales all dimensions of resolution, width, and depth. This approach ensures EfficientNet models to acquire higher accuracy and efficiency compared to previous models [24].

The EfficientNet family comprises models named from EfficientNetB0 to EfficientNetB7, each increasing in size and complexity. EfficientNetB0 is the smallest and simplest model in the family, serving as the base model. It has approximately 5.3 million parameters and performs about 0.39 billion floating-point operations (FLOPs). EfficientNetB1 is slightly larger than EfficientNetB0, with around 7.9 million parameters and approximately 0.70 billion FLOPs. EfficientNetB2 continues this trend, being larger than EfficientNetB1, with roughly 9.2 million parameters and about 1.0 billion FLOPs. EfficientNetB3 increases the size and complexity even further, containing around 12 million parameters and performing approximately 1.8 billion FLOPs. EfficientNetB4, significantly larger than B3, has around 19 million parameters and about 4.2 billion FLOPs. EfficientNetB5 is larger still, with approximately 30 million parameters and 9.9 billion FLOPs, offering very high accuracy suitable for high-end applications with substantial computational resources. EfficientNetB6, larger than B5, contains around 43 million parameters and performs approximately 19 billion FLOPs. It provides extremely high accuracy, making it suitable for applications demanding the highest performance and having ample computational power. At the top of the scale is EfficientNetB7, the largest model in the family, with around 66 million parameters and approximately 37 billion FLOPs [24].

The fundamental building block used in EfficientNet models is the MBConv block, which stands for Mobile Inverted Bottleneck Convolution [26]. This block is optimized for both performance and efficiency, making it well-suited for deep neural networks used in mobile and resource-constrained environments. EfficientNet models use a varying number of MBConv blocks in their models, depending on the specific model variant (B0 to B7). The number of MBConv blocks increases with the model size, contributing to the depth and complexity of the network [24,26].

The MBConv block combines several methods to achieve high computational efficiency and powerful feature extraction, as shown in Figure 3. These methods in the structure of the MBConv block are as follows.

Inverted Bottleneck: Traditional bottleneck layers in neural networks reduce the dimensionality of the data before processing it further. In contrast, the inverted bottleneck layer in MBConv expands the dimensionality of the input features (i.e., increases the number of channels) before applying further convolutions. This expansion allows the network to capture more complex features and then compress them back to a lower-dimensional space efficiently [27].

Depthwise Convolution (DC): This convolution operation applies a single filter to each input channel separately, as opposed to using multiple filters across all channels. This significantly reduces the computational complexity and the number of parameters required [28].

Pointwise Convolution (PC) (1x1 Convolution): After the DC, a PC is used to combine the outputs of the DC. This step effectively mixes information across different channels and compensates for the reduced computational complexity of the DC [28].

Squeeze-and-Excitation (SE) Block: The SE block is integrated within the MBConv block to enhance the representational power of the network. It consists of two main steps: *Squeeze:* Global average pooling is applied to each channel of the feature map to produce a channel descriptor, reducing each feature map to a single value. *Excitation:* These descriptors are passed through two fully connected (FC) layers with a Swish activation in between, generating a set of weights that are used to scale the original input channels. This process helps the network focus on the most important features by re-calibrating the feature maps [29].

Residual Connections: Residual connections are used to add the input of the MBConv block directly to its output. This skip connection helps in mitigating the vanishing gradient problem, making it easier to train deeper networks. It also enables the network to learn residual mappings, which are often more straightforward to optimize than learning the complete transformation [25].

The working steps of the MBConv block given in Figure 3 are as follows:

- The process starts with an input tensor, which is a set of features with a certain number of channels. The first step is to expand this input tensor by increasing the number of channels. This is done using a pointwise convolution (1x1 convolution). This expansion allows the network to capture more detailed and complex features from the input data.
- After expanding the number of channels, the next step is to apply a DC. Unlike a regular convolution that operates across all channels, a DC applies a separate filter to each channel. This means that each channel is processed independently. This step is computationally efficient because it decreases the number of parameters and operations compared to a full convolution.
- The output from the DC is then passed through a squeeze-and-excitation block.
- After the SE block has re-weighted the channels, another PC (1x1 convolution) is implemented to decrease the number of channels back to the original count. This step compresses the expanded and processed features back to a manageable number of channels, making the output tensor easier to handle.
- Dropout is used in MBConv blocks to prevent overfitting and improve generalization by randomly deactivating a subset of neurons during training.
- If the dimensions of the input tensor and the output tensor after the compression phase (second PC) are the same, a residual connection is used. This means the original input tensor is added to the output tensor, helping the network learn better by preserving the original input information.
- Batch normalization (BN) is performed after each convolution operation in the network. BN is used to stabilize and speed up the training process of deep neural networks in MBConv blocks. In addition, the activation function used in this network is Swish. The EfficientNet model uses the Swish activation function, which has been shown to improve performance compared to traditional activation functions like ReLU (Rectified Linear Unit).

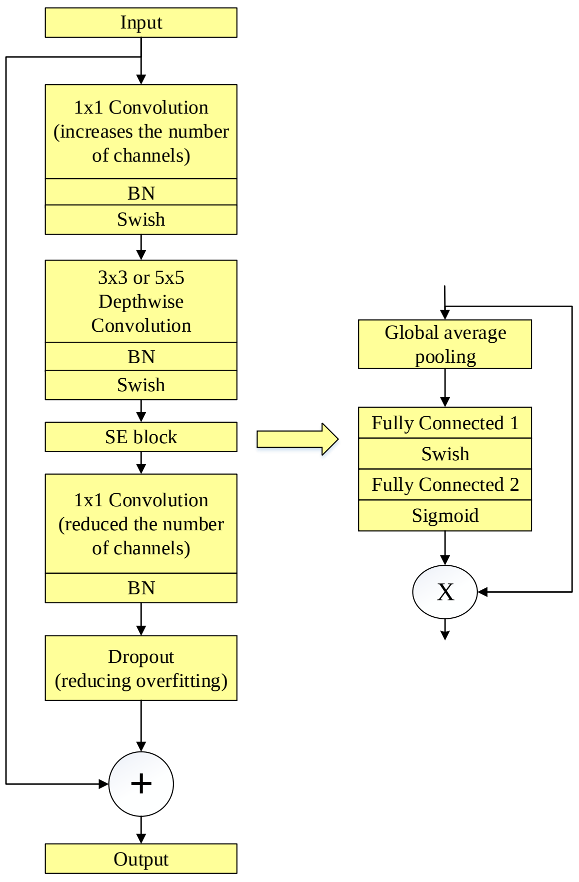

Input
1x1 Convolution (increases the number of channels)
BN
Swish
3x3 or 5x5 Depthwise Convolution
BN
Swish
SE block
1x1 Convolution (reduced the number of channels)
BN
Dropout (reducing overfitting)
+
Output
Global average pooling
Fully Connected 1
Swish
Fully Connected 2
Sigmoid
X

Figure 3. Structure of MBConv

2.5. Proposed Model

In this study, a concatenation of ResNet and EfficientNet models is proposed for automated classification of faults in solar panels. The proposed model is a concatenation of ResNet101 and EfficientNetB1 models as shown in Figure 4. The reasons for using ResNet101 in the proposed model are as follows: (1) ResNet101's depth allows it to capture more detailed features than ResNet50, making it more effective for tasks that require distinguishing between fine-grained classes. (2) ResNet101 strikes a balance between the number of layers and computational efficiency, making it suitable for tasks that are not too simple but also not extremely complex. (3) Compared to ResNet152, ResNet101 is less prone to overfitting on small datasets, which is crucial for maintaining good generalization performance. In summary, ResNet101 outperformed ResNet50 and ResNet152 because it provides the right depth and complexity for the task at hand. It is deep enough to capture complex patterns in the solar panel images, but not so deep that it overfits on the relatively small dataset, making it the best choice for achieving high accuracy. The reasons for using the EfficientNetB1 model in the proposed model are as follows: (1) EfficientNetB1 has more parameters and a higher resolution input size compared to EfficientNetB0, allowing it to capture more detailed features and improve classification accuracy. (2) Compared to EfficientNetB0, The additional layers and width in EfficientNetB1 help in learning more complex patterns and subtle differences, which is crucial for tasks like fault detection in solar panels where fine details matter. (3) EfficientNetB1 has significantly fewer parameters and FLOPs compared to larger models (EfficientNetB2 to B7), making it more efficient and suitable for environments with limited computational resources.

The proposed model starts with an input image. The size of the input solar panel image is 224 x 224 x 3. Then, this input image is given to the input of ResNet101 and EfficientNetB1 models separately and feature extraction is performed. A detailed description of the layer steps of the ResNet101 model when performing feature extraction on the 224x224x3 input image is as follows. Following the input layer, the image is passed through the first convolutional layer, which applies 64 filters, each of size 7x7, with a stride of 2. This operation increases the depth of the feature maps and reduces the spatial dimensions, resulting in an output of size 112x112x64. This initial convolutional layer is crucial for capturing low-level features, such as edges and textures, in the input image. The output from the convolutional layer is then normalized using a batch normalization layer, which stabilizes and accelerates the training process by normalizing the inputs to each layer. After batch normalization, a ReLU activation function is implemented to introduce non-linearity into the model, enabling it to learn more complex representations. Next, a max pooling layer with a filter size of 3x3 and a stride of 2 further reduces the spatial dimensions, yielding an output of size 56x56x64. This pooling operation helps to down-sample the feature maps and reduce computational complexity, while also retaining important features. ResNet101 is primarily built using bottleneck residual blocks, each consisting of three convolutional layers: 1x1, 3x3, and 1x1. The network contains several groups of these blocks, organized to gradually reduce the spatial dimensions while increasing the depth, allowing the network to learn increasingly abstract features. The bottleneck residual blocks in the ResNet101 model are as follows:

Conv2x Stage (3 Blocks): The first set of residual blocks, known as Conv2x, begins with a bottleneck block that comprises three key layers: 1x1 Convolution Layer uses 64 filters to reduce the dimensionality of the input, making the computation more efficient. 3x3 Convolution Layer (the main processing layer), which employs 64 filters to perform convolutions and extract features. 1x1 Convolution Layer restores the dimensionality using 256 filters, preparing the output for the next stage. Within each bottleneck block, a residual connection adds the input to the output, creating a shortcut path that enables gradients to flow more easily during backpropagation. This mechanism is central to the success of residual networks, as it alleviates the vanishing gradient problem. The Conv2_x stage contains three of these bottleneck blocks, each maintaining an output size of 56x56x256.

Conv3x Stage (4 Blocks): The next stage, Conv3x, introduces the first bottleneck block with downsampling, which reduces the spatial dimensions while increasing the depth. 1x1 Convolution Layer utilizes 128 filters to reduce dimensions. 3x3 Convolution Layer continues processing with 128 filters. 1x1 Convolution Layer increases dimensions using 512 filters. The Conv3_x stage consists of four bottleneck blocks, resulting in an output size of 28x28x512.

Conv4x Stage (23 Blocks): The Conv4x stage is the deepest stage, consisting of 23 bottleneck blocks, each further refining the feature maps. 1x1 Convolution Layer employs 256 filters for dimensionality reduction. 3x3 Convolution Layer applies 256 filters for feature extraction. 1x1 Convolution Layer restores dimensions with 1024 filters. The Conv4x stage maintains an output size of 14x14x1024.

Conv5x Stage (3 Blocks): The final set of blocks, Conv5x, consists of three bottleneck blocks that further distill the features. 1x1 Convolution Layer uses 512 filters for dimensionality reduction. 3x3 Convolution Layer processes with 512 filters. 1x1 Convolution Layer expands dimensions with 2048 filters. The output from this stage has a size of 7x7x2048. After the final convolutional stage, ResNet101 employs a global average pooling (GAP) layer. This layer reduces each feature map to a single value by averaging, resulting in a 2048-dimensional feature vector. This vector effectively summarizes the learned features from the input image.

A detailed description of the layer steps of the EfficientNetB1 model when performing feature extraction on the 224x224x3 input image is as follows. Following the input layer, the image is passed through the first convolutional layer, which applies 32 filters, each of size 3x3, with a stride of 2. The output size obtained at the end of this process is 112x112x32. MBConv blocks in the EfficientNetB1 model are applied to the obtained output feature map. EfficientNetB1 consists of seven MBConv blocks. These are depth separable convolutions that are efficient and help to decrease the number of parameters and computations while maintaining performance. The blocks are organised as follows: Block 1 includes two MBConv 3x3 layers. Block 2 includes three MBConv 3x3 layers. Block 3 includes three MBConv 5x5 layers. Block 4 includes four MBConv 3x3 layers. Block 5 includes four MBConv 5x5 layers. Block 6 includes five MBConv 5x5 layers. Block 7 includes two MBConv 3x3 layers. The output from all MBConv block has a size of 7x7x2560. After the final MBConv block, EfficientNetB1 employs the GAP layer. This layer reduces each feature map to a single value by averaging, resulting in a 2560-dimensional feature vector. After performing feature extraction using both ResNet101 and EfficientNetB1 models, the obtained feature maps are passed through a GAP layer, and then the two models are concatenated. The concatenated model results in a 4608-dimensional feature vector. Subsequently, a FC layer with 128 neurons is applied. After the FC layer, batch normalization is applied, followed by a dropout layer with a dropout rate of 0.3. Finally, classification is performed using a softmax classifier.

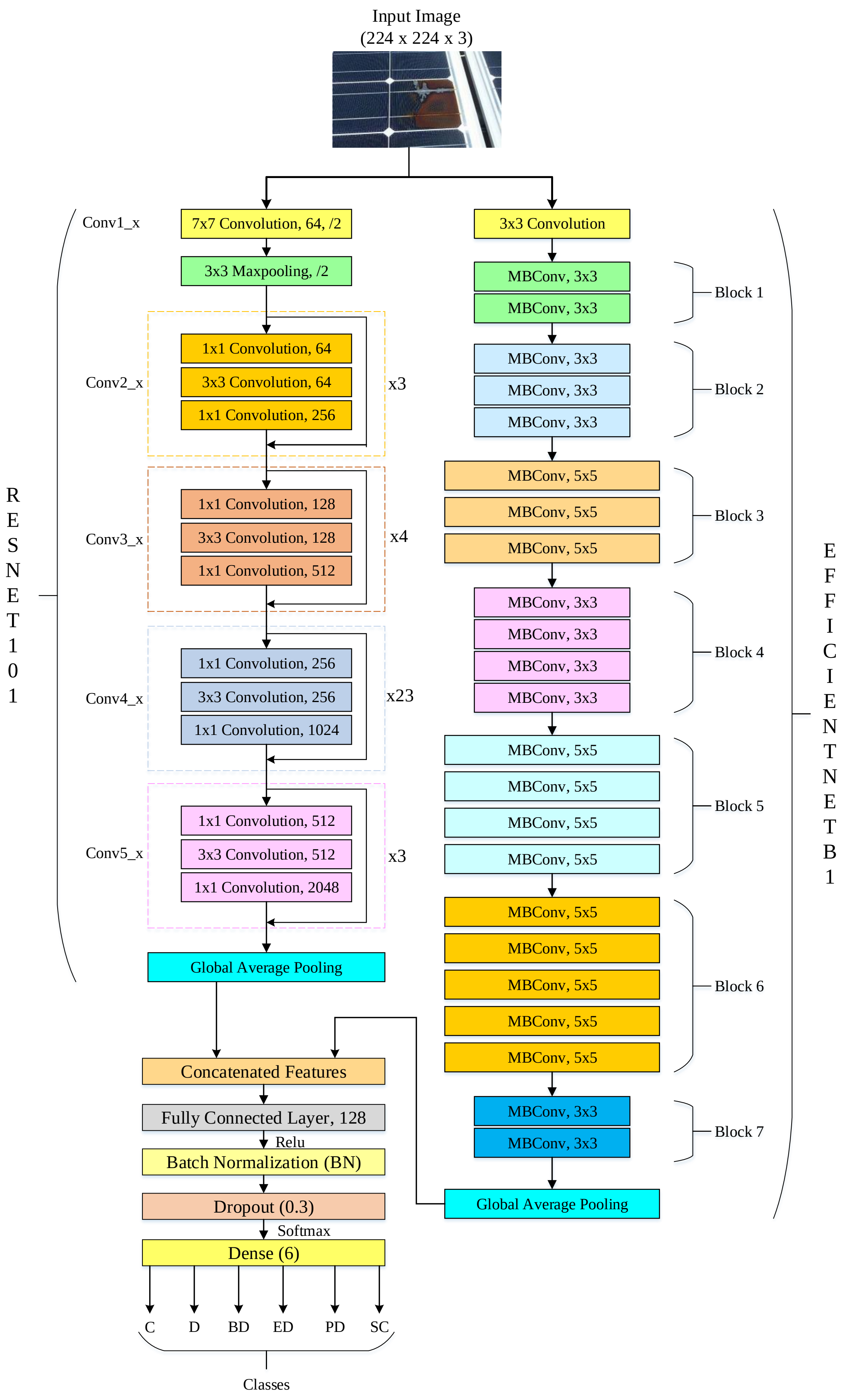


Figure 4. Structure of proposed model

3. EXPERIMENTS

3.1. Experimental Setup

In the experimental studies, the python programming language and Keras-TensorFlow libraries were used. All python code was written on the Kaggle notebook platform. For training the proposed model on the Kaggle platform, a GPU P100 was used as the execution environment. The hyperparameters used are as follows: batch size 64, learning rate 0.0001, and the number of epochs was set to 150. The Adam optimization method was used. The input image dimensions were set to 224x224x3. The dataset containing 885 solar panel images was split into training and test datasets as 70% and 30%, respectively. That is, 620 images were used for training and 265 images for testing. Sparse Categorical Crossentropy (SCC) loss function is used to train the proposed model. The SCC is a loss function that is particularly advantageous for multi-class classification problems where labels are available as integers rather than one-hot encoded vectors. This loss function is efficient because it directly uses integer labels, reducing memory and computational overhead compared to Categorical Crossentropy, which requires one-hot encoded labels. It is well-suited for tasks where each instance belongs to a single class among many, providing a straightforward probabilistic interpretation of predictions through integration with the softmax activation function. This efficiency and compatibility make the SCC an ideal choice for training models with a significant number of classes, as in our solar panel image classification task [30-32].

3.2. Evaluation Metrics

Different evaluation metrics were used to compare the performance of the proposed model. The following four metrics are commonly used when observing various criteria of a classifier.

Accuracy represents the ratio of correct predictions to the total number of predictions made by the model. Accuracy is calculated as in Equation (1) [33].

$$Accuracy\ (Acc) = \frac{TP + TN}{TP + FP + TN + FN} \tag{1}$$

In the context of multi-class classification, precision is a metric that evaluates the accuracy of positive predictions made by a model for each class. Precision measures how many of the examples predicted as positive actually belong to the class. It is defined as shown in Equation (2) [33].

$$Precision\ (P) = \frac{TP}{TP + FP} \tag{2}$$

Recall is a metric used in classification to evaluate a model's ability to correctly identify all relevant examples (true positives) from the total actual positive examples (true positives + false negatives). In other words, it measures the percentage of true positive examples correctly identified by the model. It is defined as shown in Equation (3) [33].

$$Recall\ (R) = \frac{TP}{TP + FN} \tag{3}$$

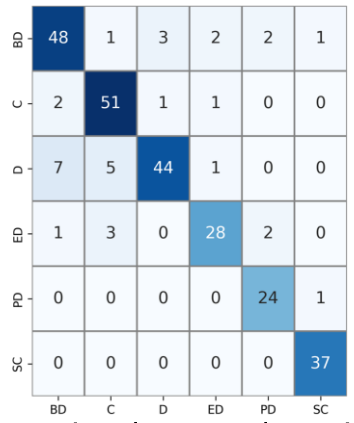
The F1-score is a metric used in multi-class classification to balance precision and recall. By combining both precision and recall into a single value, it becomes a useful metric for evaluating the overall performance of a method. The F1-score is calculated as the harmonic mean of precision and recall, and it is particularly valuable when a balance between false positives and false negatives is desired. It is calculated as shown in Equation (4) [33].

$$F1 - score\ (F1s) = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

The terms TN, TP, FN, and FP in Equations (1)-(4) are derived from the confusion matrix. True Negatives (TN) represent the number of examples that the model correctly predicts as belonging to the negative class (e.g., "0" or "no"). These are cases where the model correctly identifies examples as not belonging to the positive class when they truly don't. True Positives (TP) represent the number of examples that the model correctly predicts as belonging to the positive class (e.g., "1" or "yes"). These are cases where the model correctly identifies examples as belonging to the positive class when they truly do. False Positives (FP) represent the number of examples that are actually in the negative class but are incorrectly predicted by the model as belonging to the positive class. These are situations where the model makes a positive prediction when it should have made a negative one. False Negatives (FN) represent the number of examples that are actually in the positive class but are incorrectly predicted by the model as belonging to the negative class. These are situations where the model makes a negative prediction when it should have made a positive one [33,34].

3.3. Experimental Results and Discussion

The proposed model is a concatenation of ResNet and EfficientNet models. In the proposed model, the best classification result is obtained with the concatenation of ResNet101 and EfficientNetB1 models. The confusion matrix of the proposed model (ResNet101 + EfficientNetB1) is given in Figure 5. When examining Figure 5, it can be seen that 48 out of a total of 57 panel images in class Bird-drops (BD) are correctly classified. Similarly, 51 out of a total of 55 images in class Clean (C), 44 out of 57 images in class Dusty (D), 28 out of 34 images in class Electrical-damage (ED), 24 out of 25 images in class Physical-damage (PD), and finally, all 37 Snow-covered (SC) images are correctly classified. Out of a total of 265 test images, 232 are correctly classified. In this case, the test accuracy is (232/265) * 100 = 87.55%.



	BD	C	D	ED	PD	SC
BD	48	1	3	2	2	1
C	2	51	1	1	0	0
D	7	5	44	1	0	0
ED	1	3	0	28	2	0
PD	0	0	0	0	24	1
SC	0	0	0	0	0	37

Figure 5. The confusion matrix of proposed model

In this study, comparisons were made with various pre-trained deep learning models including EfficientNetB0-B7 [24], ResNet50-101-152 [25], VGG16-19 [35], MobileNet [36], MobileNetV2 [27], DenseNet121-169-201 [37] to classify faults in solar panel images. The results of the comparison are given in Table 1. When Table 1 is analyzed, it is clearly seen that the best results are obtained with ResNet and EfficientNet models. Table 1 also shows the classification results obtained by combining ResNet50-101-201 and EfficientNetB0-B7 models. A detailed analysis of all the models in Table 1 is as follows:

The VGG16 and VGG19 models, known for their simplicity and depth, achieved moderate performance with VGG16 slightly outperforming VGG19. VGG16 achieves an Acc of 66.79%, with P of 69.12%, R of 67.82%, and an F1s of 68.24%. VGG19, on the other hand, has slightly lower metrics across the board, with an Acc of 65.28%, P of 67.50%, R of 65.96%, and an F1s of 66.20%. The marginally better performance of VGG16 suggests that, for this dataset, the additional depth of VGG19 does not translate into improved results. This may be due to the increased complexity and overfitting associated with deeper networks.

MobileNet models are designed for efficiency, balancing performance with computational cost. The original MobileNet achieves an Acc of 66.42%, with P, R, and F1s around 65%. MobileNet performed similarly to VGG16 with an Acc of 66.42%. However, MobileNetV2 shows a notable drop in performance, with all metrics hovering around 59-60%. This decline suggests that the architectural changes in MobileNetV2, which aim to improve efficiency, may have compromised its ability to capture relevant features in this specific dataset.

DenseNet models, known for their dense connectivity, generally outperform VGG and MobileNet models. DenseNet121, DenseNet169, and DenseNet201 exhibit accuracies of 69.43%, 69.06%, and 68.30%, respectively. P, R, and F1s for these models are also consistently high, can indicating reliable performance. These results show that DenseNet architectures are more capable of capturing complex images in data than VGG and MobileNet models. The improvement over VGG and MobileNet models can be attributed to the enhanced feature propagation and reduced vanishing-gradient problem inherent in DenseNet architectures.

ResNet models significantly outperform the previously discussed models (VGG, MobileNet, and DenseNet). ResNet50 achieves an Acc of 79.62%, with P and F1s both at 80.40%, and a R of 80.50% [38]. Larger ResNet models, such as ResNet101 and ResNet152, further enhance performance, with ResNet152 achieving the highest individual model Acc of 84.15%, P of 85.72%, R of 82.84%, and an F1s of 83.97% [38]. In addition, ResNet101 achieving individual model Acc of 83.40%, P of 84.37%, R of 84.04%, and an F1s of 84.07% [38]. The success of ResNet architectures can be attributed to their ability to mitigate the vanishing gradient problem through skip connections, allowing them to maintain high performance even with increased depth.

EfficientNet models also exhibit strong performance, with several variants outperforming most other EfficientNet models. EfficientNetB2 and EfficientNetB4 stand out with accuracies of 82.64% and 83.40%, respectively. These models also maintain high precision, recall, and F1-scores, indicating balanced performance across different metrics. The EfficientNet model scales depth, width, and resolution in a compound manner, optimizing performance while maintaining computational efficiency.

The concatenation of ResNet and EfficientNet models yields the highest performance metrics. Notably, the concatenation of ResNet101 and EfficientNetB1 (proposed model) achieves the highest overall performance, with an Acc of 87.55%, P of 87.92%, R of 88.75%, and F1s of 88.13%. This concatenation leverages the strengths of both models—ResNet's robust feature learning and EfficientNet's balanced scaling—resulting in superior classification capabilities. Other successful concatenation include ResNet101 with EfficientNetB3, ResNet101 with EfficientNetB4, ResNet101 with EfficientNetB6, and ResNet152 with EfficientNetB5, all of which show high accuracies and balanced metric scores. These results suggest that model ensembling, particularly concatenating different models, can effectively enhance performance by capturing diverse feature representations.

TABLE I
COMPARISON WITH DIFFERENT DEEP LEARNING MODELS

Models	Acc(%)	P(%)	R(%)	F1s(%)
VGG16	66.79	69.12	67.82	68.24
VGG19	65.28	67.50	65.96	66.20
MobileNet	66.42	65.56	65.84	65.64
MobileNetV2	59.62	59.54	59.59	59.27
DenseNet121	69.43	70.99	68.17	69.19
DenseNet169	69.06	69.91	69.47	69.49
DenseNet201	68.30	70.38	66.85	67.86
ResNet50	79.62	80.40	80.50	80.40
ResNet101	83.40	84.37	84.04	84.07
ResNet152	84.15	85.72	82.84	83.97
EfficientNetB0	81.13	82.39	82.16	82.07
EfficientNetB1	81.89	83.39	81.36	82.06
EfficientNetB2	82.64	83.96	83.54	83.67
EfficientNetB3	80.00	81.76	80.70	81.09
EfficientNetB4	83.40	83.68	83.45	83.41
EfficientNetB5	81.89	82.81	81.96	82.26
EfficientNetB6	77.74	78.75	78.56	78.50
EfficientNetB7	76.60	78.02	76.90	77.31
ResNet50 + EfficientNetB0	83.02	83.11	84.13	83.53
ResNet50 + EfficientNetB1	82.64	84.17	83.87	83.96
ResNet50 + EfficientNetB2	83.02	83.64	83.41	83.37
ResNet50 + EfficientNetB3	80.76	81.47	81.06	81.12
ResNet50 + EfficientNetB4	81.51	81.71	82.60	82.10
ResNet50 + EfficientNetB5	83.77	85.37	85.47	85.28
ResNet50 + EfficientNetB6	82.26	83.55	82.25	82.71
ResNet50 + EfficientNetB7	83.02	84.18	84.51	84.29
ResNet101 + EfficientNetB0	83.40	84.37	84.62	84.28
ResNet101 + EfficientNetB1	**87.55**	**87.92**	**88.75**	**88.13**
ResNet101 + EfficientNetB2	86.42	86.24	86.11	86.11
ResNet101 + EfficientNetB3	87.17	**88.49**	87.58	87.86
ResNet101 + EfficientNetB4	87.17	87.24	87.23	87.21
ResNet101 + EfficientNetB5	85.28	85.82	86.09	85.76
ResNet101 + EfficientNetB6	87.17	88.03	86.87	87.32
ResNet101 + EfficientNetB7	84.53	85.13	84.40	84.53
ResNet152 + EfficientNetB0	86.04	87.23	86.47	86.78
ResNet152 + EfficientNetB1	83.77	86.32	84.12	84.89
ResNet152 + EfficientNetB2	83.77	84.23	83.06	83.50
ResNet152 + EfficientNetB3	85.66	87.76	86.71	87.12
ResNet152 + EfficientNetB4	85.28	85.64	85.30	85.44
ResNet152 + EfficientNetB5	86.79	87.02	87.02	86.96
ResNet152 + EfficientNetB6	81.51	82.15	80.96	81.40
ResNet152 + EfficientNetB7	84.15	85.42	84.80	85.07

4. CONCLUSION

SPs play a crucial role in the global transition to renewable energy, offering a sustainable solution to meet the world's energy demands. As a clean and abundant energy source, SPs contribute to reducing carbon emissions and combating climate change. However, the effectiveness and reliability of solar energy systems can be significantly impacted by faults and defects in SPs. Common issues such as cracks, dusty, snow-covered, soiling, and shading can reduce the efficiency of solar panels, leading to decreased energy output and increased maintenance costs. Therefore, accurate and timely detection of these faults is essential for maintaining the performance and longevity of solar energy systems.

In this study, we propose a hybrid DL model using ResNet and EfficientNet models to classify faults in solar panels. ResNet is renowned for its ability to train very deep neural networks effectively, utilizing residual connections to prevent issues like vanishing gradients. This capability allows ResNet to capture intricate patterns and features in complex datasets, which is essential for detecting subtle defects in solar panels. On the other hand, EfficientNet is designed to achieve a balance between accuracy and computational efficiency by systematically scaling the network's resolution, width, and depth. By integrating ResNet and EfficientNet models, our approach benefits from the strengths of both architectures: the depth and learning capacity of ResNet and the optimized performance of EfficientNet.

The experimental results demonstrate that the combined ResNet101 + EfficientNetB1 model significantly outperforms individual models in terms of Acc, P, R, and F1s. This hybrid model achieved an Acc of 87.55%, P of 87.92%, R of 88.75%, and F1s of 88.13%, marking notable improvements over the closest models. The synergistic use of ResNet and EfficientNet enables the proposed model to accurately identify and classify various faults in solar panels, thereby enhancing the reliability and efficiency of solar energy systems.

In conclusion, the combination of ResNet and EfficientNet models offers a powerful solution for detecting and classifying faults in solar panels. This approach not only improves fault detection accuracy but also contributes to the overall performance and sustainability of solar energy systems. As the demand for solar energy continues to rise, implementing advanced deep learning models like the one proposed in this study will be essential for ensuring the long-term viability and efficiency of solar power installations.

In future work, the primary aim is to further enhance the model's performance by experimenting with cutting-edge neural network models such as Vision Transformers and different ensemble methods. Additionally, research is planned to improve the model's ability to detect and classify faults by incorporating additional data sources, such as thermal imaging and real-time monitoring data.

REFERENCES

[1] D. Korkmaz and H. Acikgoz, "An efficient fault classification method in solar photovoltaic modules using transfer learning and multi-scale convolutional neural network," *Eng. Appl. Artif. Intell.*, vol. 113, no. April, p. 104959, 2022.

[2] H. Acikgoz, "A novel approach based on integration of convolutional neural networks and deep feature selection for short-term solar radiation forecasting," *Appl. Energy*, vol. 305, no. June 2021, p. 117912, 2022.

[3] T. Z. Ang, M. Salem, M. Kamarol, H. S. Das, M. A. Nazari, and N. Prabaharan, "A comprehensive study of renewable energy sources: Classifications, challenges and suggestions," *Energy Strateg. Rev.*, vol. 43, no. November 2021, p. 100939, 2022.

[4] B. Li, C. Delpha, D. Diallo, and A. Migan-Dubois, "Application of Artificial Neural Networks to photovoltaic fault detection and diagnosis: A review," *Renew. Sustain. Energy Rev.*, vol. 138, no. October 2020, 2021.

[5] A. Rico Espinosa, M. Bressan, and L. F. Giraldo, "Failure signature classification in solar photovoltaic plants using RGB images and convolutional neural networks," *Renew. Energy*, vol. 162, pp. 249–256, 2020.

[6] M. Le, L. Van Su, N. Dang Khoa, V. D. Dao, V. Ngoc Hung, and V. Hong Ha Thi, "Remote anomaly detection and classification of solar photovoltaic modules based on deep neural network," *Sustain. Energy Technol. Assessments*, vol. 48, no. June, p. 101545, 2021.

[7] C. Haydaroğlu, H. Kılıç, and B. Gümüş, "Performance Analysis and Comparison of Performance Ratio of Solar Power Plant," *Turkish J. Electr. Power Energy Syst.*, vol. 4, pp. 190–199, 2024.

[8] H. KILIC, M. YILMAZ, and B. GUMUS, "Fault Detection in Photovoltaic Arrays: a Robust Regularized Machine Learning Approach," *Dyna*, vol. 95, no. 1, pp. 622–628, 2020.

[9] K. Osmani, A. Haddad, T. Lemenand, B. Castanier, M. Alkhedher, and M. Ramadan, "A critical review of PV systems' faults with the relevant detection methods," *Energy Nexus*, vol. 12, no. September, p. 100257, 2023.

[10] G. R. Venkatakrishnan *et al.*, "Detection, location, and diagnosis of different faults in large solar PV system—a review," *Int. J. Low-Carbon Technol.*, vol. 18, no. 1, pp. 659–674, 2023.

[11] Z. B. Duranay, "Fault Detection in Solar Energy Systems: A Deep Learning Approach," *Electron.*, vol. 12, no. 21, 2023.

[12] A. Mahmud, M. S. R. Shishir, R. Hasan, and M. Rahman, "A comprehensive study for solar panel fault detection using VGG16 and VGG19 convolutional neural networks," *2023 26th Int. Conf. Comput. Inf. Technol. ICCIT 2023*, pp. 1–6, 2023.

[13] M. M. Taye, "Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions," *Computers*, vol. 12, no. 5, 2023.

[14] S. H. Han, T. Rahim, and S. Y. Shin, "Detection of faults in solar panels using deep learning," *2021 Int. Conf. Electron. Information, Commun. ICEIC 2021*, pp. 2–5, 2021.

[15] W. Tang, Q. Yang, K. Xiong, and W. Yan, "Deep learning based automatic defect identification of photovoltaic module using electroluminescence images," *Sol. Energy*, vol. 201, no. November 2019, pp. 453–460, 2020.

[16] S. Naveen Venkatesh and V. Sugumaran, "Fault Detection in aerial images of photovoltaic modules based on Deep learning," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1012, no. 1, p. 012030, 2021.

[17] G. S. Eldeghady, H. A. Kamal, and M. A. M. Hassan, "Fault diagnosis for PV system using a deep learning optimized via PSO heuristic combination technique," *Electr. Eng.*, vol. 105, no. 4, pp. 2287–2301, 2023.

[18] R. H. Fonseca Alves, G. A. de Deus Júnior, E. G. Marra, and R. P. Lemos, "Automatic fault classification in photovoltaic modules using Convolutional Neural Networks," *Renew. Energy*, vol. 179, pp. 502–516, 2021.

[19] S. H. Lee, L. C. Yan, and C. S. Yang, "LIRNet: A Lightweight Inception Residual Convolutional Network for Solar Panel Defect Classification," *Energies*, vol. 16, no. 5, pp. 1–12, 2023.

[20] Afroz, "Solar Panel Images Clean and Faulty Images," *Kaggle*, 2023. [Online]. Available: https://www.kaggle.com/datasets/pythonafroz/solar-panel-images. [Accessed: 10-May-2024].

[21] A. W. Salehi *et al.*, "A Study of CNN and Transfer Learning in Medical Imaging: Advantages, Challenges, Future Scope," *Sustainability*, vol. 15, no. 7, 2023.

[22] N. Raza, A. Naseer, M. Tamoor, and K. Zafar, "Alzheimer Disease Classification through Transfer Learning Approach," *Diagnostics*, vol. 13, no. 4, 2023.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 248–255, 2010.

[24] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, pp. 770–778.

[26] B. Baheti, S. Innani, S. Gajre, and S. Talbar, "Eff-UNet: A novel architecture for semantic segmentation in unstructured environment," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 2020-June, no. September 2021, pp. 1473–1481, 2020.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4510–4520,

2018.

[28] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1800–1807, 2017.

[29] H. Fırat, "Classification of White Blood Cells using the Squeeze-Excitation Residual Network," *Bilişim Teknol. Derg.*, vol. 16, no. 3, pp. 189–205, 2023.

[30] B. N. Chaithanya, T. J. Swasthika Jain, A. Usha Ruby, and A. Parveen, "An approach to categorize chest X-ray images using sparse categorical cross entropy," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 24, no. 3, pp. 1700–1710, 2021.

[31] P. Naveen, "Phish: A novel hyper-optimizable activation function," *techrxiv.orgP NaveenAuthorea Prepr. 2023•techrxiv.org*, pp. 1–8, 2023.

[32] A. Bhat, A. V. Krishna, and S. Acharya, "Analytical Comparison of Classification Models for Raga Identification in Carnatic Classical Instrumental Polyphonic Audio," *SN Comput. Sci.*, vol. 1, no. 6, pp. 1–9, 2020.

[33] H. Dalianis, "Evaluation Metrics and Evaluation," *Clin. Text Min.*, no. 1967, pp. 45–53, 2018.

[34] S. A. Hicks *et al.*, "On evaluation metrics for medical applications of artificial intelligence," *Sci. Rep.*, vol. 12, no. 1, pp. 1–9, 2022.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[36] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 2017.

[38] R. Akınca, H. Fırat, and M. E. Asker, "Transfer Öğrenme Tabanlı ResNet Modeli Kullanılarak Güneş Panellerindeki Hataların Tespiti," *Dicle Üniversitesi 2.Uluslararası Fen Bilim. Lisansüstü Araştırmalar Sempozyumu*, pp. 27–30, 2024.

BIOGRAPHIES

Rojbin Akınca received her Bsc. degree in electrical and electronics engineering from Dicle University, Diyarbakır, Turkey in 2018. She continues her MSc. degree in Renewable Energy Resources at Dicle University as of 2023. She is currently working as a Data Preparation and Control Operator at Dicle University. Her research interests include renewable energy resources, artificial intelligence, deep learning and machine learning.

Hüseyin Fırat received the BSc. degree in computer engineering from Cukurova University, Adana, Turkey, in 2014. He received the MSc. degree in computer engineering from Inonu University in 2018. He received the PhD degree in computer engineering from Inonu University, Turkey, in 2022. He also works as assistant professor at Dicle University in Turkey. His current interests include remote sensing, deep learning, pattern recognition, signal processing, medical image processing and image classificaiton.

Mehmet Emin Asker received the BSc. degree in electrical electronics engineering, from Firat University, Elazig, Turkey in 1997, the MSc. degree and the PhD degree in electrical machines, power electronics from Firat University, Elazig, Turkey, in 2009 and 2016, respectively. He is an assistant professor with Dicle University, department of electrical power and energy. Where he teaches courses on power system, power electronics, circuit theory and electrical machines since 2007. His research interests include electrical machines, power electronics, chaos, machine learning and power systems.