

# Advancing Solar Cell Analysis: Leveraging Computer Vision for Electroluminescence Imaging in Solar Panels

By Team BallsDeepLearning for COMP9517

Haoming Zhou  
z5447385

James White  
z5363399

Hyeongjun Kim  
z5311658

Abhijeet Agarwal  
z5486197

Harsh Bhatia  
z5363556

## I. INTRODUCTION

Solar panels are widely used as a means of producing renewable and clean energy. They consist of photovoltaic (PV) cells that generate an electric current when exposed to sunlight. There are a multitude of factors that can compromise the longevity and performance of photovoltaic cells. The integrity of the cells are susceptible to the mechanical impact of hail and falling tree branches and over time the material can degrade due to exposure to environmental conditions such as temperature and sunlight. The manufacturing process is also a potential cause of defects and damages.

As damaged and defective cells can be less efficient, the continual inspection and maintenance of panels are necessitated. The conventional inspection methods through human experts are flawed, as they can not be scaled or are inaccurate. Some material cell defects are unable to be detected by the human eye, while some detected features have no impact on performance.

Electroluminescence (EL) imaging is a technology that offers high-fidelity visualization of the cell's condition by inducing EL emissions. With EL techniques, imperfections and defects can be screened as images which provides a more accurate inspection.

The goal of our project is to implement advanced computer vision methods to analyse EL images, and provide an automated and scalable solution to providing precise predictions on the health of PV cells.

This project will utilise the ELPV dataset, which consists of 2,624 samples of 300x300 pixels grayscale images of functional and defective solar cells with varying degrees of degradation extracted from 44 different solar modules [1]. An example of a solar cell image in the dataset is shown in figure 1.

Each image of a PV cell in the ELPV dataset is paired with one of four probability labels, 0, 0.33, 0.67, and 1, which indicate the probability that the cell is defective. In our report, these labels correspond respectively to 'fully functional', 'possibly defective', 'likely defective', and 'certainly defective'. Across the 2,624 samples, there are 1508 samples labeled 'fully functional', 295 samples labeled 'possibly defective', 106 samples labeled 'likely defective', and 715

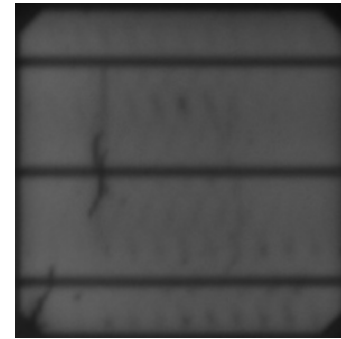


Fig. 1. ELPV Image Sample

samples labeled 'certainly defective'. We observe that there is a clear class imbalance in our dataset, where there are significantly more samples that are 'fully functional' and 'certainly defective' than there are samples in between those classes.

Each image is also labeled with the 'type' of solar module, 'mono' for monocrystalline, and 'poly' for polycrystalline. There are 1074 samples of 'mono' cells and 1550 samples of 'poly' cells. From visual inspection, the distinctions between these types are not immediately obvious.

By observing the images of both healthy and defective cells, we found several features and heuristics that were likely to correspond to defective cells. All the images of the cells consist of two or three horizontal lines, but diagonal lines potentially representing cracks were often present in defective cells. Another indication of defective cells was black regions within the mostly white cells. These regions were present as smudge-like blobs as well as the interior of polygons formed by various cracks. In the color version of the dataset, it is evident that defective cells correspond to an increasingly intense red coloration. In the grey-scale version of the dataset that is used for this group project, this corresponds to an increasing dark coloration. Given that there are relatively few samples that are 'probably defective', it is notable that the slight coloration change from 'fully functional' to 'probably defective' was an identifiable visual discriminator, for non-experts. From the ELPV dataset, we aim to build and train different models, including a basic Conv2d model

as a benchmark model, transfer learning models utilising the pre-trained ResNet-50 networks, an improved ResNet network called ResNet Plus, ShuffleNet, and ShuffleNet enhanced with an Attention Mechanism, to classify the images according to their probability of defectiveness and collect the outcome results (Part 4 *Experimental Results*). These methods will be explained in Part 3 *Methods*. Then we test and discuss the results in Part 5 *Discussion* and conclude in Part 6 *Conclusion*. For learning purposes, we built several other models for performance evaluation too, though they were not fully implemented as this report requires. These models will be added to the Part 3 *Methods* section as supplements.

## II. LITERATURE

The nature and quality of a dataset profoundly influence the accuracy and effectiveness of image classification models. In our study, since the ELPV dataset comes pre-processed with well-defined splits and accurately labeled images, we can skip the data cleaning steps and concentrate our efforts more intensively on other aspects of the machine learning pipeline. Our primary focus will be data segmentation and augmentation, model selection and architecture design, and hyperparameter tuning.

### A. Data Augmentation

Data augmentation is a technique that is used to enhance the quality and the size of the training dataset by applying transformations to the original samples and adding the transformations to the training dataset. A principal benefit of data augmentation is that it doesn't require new data and can increase the amount of training data using information only in our training data[2]. Common transformations that are used include image rotation, horizontal and vertical flipping, enhanced contrast, and the insertion of random noise. A small training dataset is a common motivation for data augmentation, especially when working with neural networks that require large amounts of data to appropriately tune their parameters. If the neural network were to train on a small dataset, it may be prone to quickly overfitting, and have poor performance. The key benefit of data augmentation is that it does not require the acquisition of new data. In addressing datasets that suffer from class imbalance, data augmentation can be used in combination with other techniques such as oversampling to provide the model with a more balanced distribution of inputs.

### B. CNN

Convolutional Neural Networks (CNN) are a class of deep neural networks that are well equipped for image-related tasks such as object detection and image classification. Compared to networks with full connectivity between adjacent layers, CNNs are easier to train and generalise better. [2]. The architecture of a CNN contains three types of layers, convolution, pooling and fully connected layers. The convolutional layer applies convolutional operations to the image data using kernels, which enables the model to extract features. Successive convolutional layers allow the network to learn spatial hierarchies, meaning

that the network is able to recognize increasingly complex features. The pooling layers reduce the dimensions of the data while retaining relevant information, which has the benefit of reducing computations, reducing overfitting and improving the translational invariance of the network.

Besides the proven effectiveness of CNNs for image-related tasks, a key benefit of CNNs over traditional computer vision methods is that it does not require hand-crafted feature extraction or segmentation by experts. This means that CNN implementations are effective and readily scaled onto domains in the absence of domain knowledge or where domain knowledge is flawed.

### C. Transfer Learning

Transfer learning is a machine learning technique where a pre-trained model developed for another task is reused to develop a new model for a new and related task. Compared to conventional machine learning methods that require large amounts of data and computational resources, transfer learning allows people to apply pre-trained models that have learned patterns and features from sizable datasets to new tasks with limited amounts of data. There are two primary types of transfer learning: Feature Extraction and Fine-Tuning. In feature extraction, the layers that are learned to detect general features on the original dataset are kept frozen(unchanged), and only the last few layers (usually for classification) of the model are retrained. In fine-tuning, the weights and parameters are modified to accustom to the new tasks[4]. For this project, we select ResNet and ShuffleNet for transfer learning.

### D. ResNet

ResNet refers to Residual Networks in deep learning. In residual networks, there are blocks called "residential blocks" that allow the input to bypass layers, unlike traditional networks where inputs must pass through each layer sequentially. It is highly effective and accurate and consumes less computational complexity in deep layers[5]. In recent years, the ResNet architecture models have evolved to a state with more layers, expanding from 18 layers to more complex versions with 50, 101, and 152 layers. For this project, we select ResNet with 50, 101, and 152 layers for transfer learning. Different models of different layers enable us to observe and analyze the impact of layer depth on the network's performance and efficiency.

### E. ShuffleNet

ShuffleNet is a convolutional neural network designed specifically for mobile and resource-constrained environments. The core parts of ShuffleNet are "group convolutions" and "the channel shuffle operation". Group convolution is a convolution technique by which the input channels are divided into groups, and each group is convolved separately[6]. The channel shuffle operation is that after dividing the input channels into groups and passing them through group convolutions, the channels are rearranged[7]. With the innovation of the channel shuffle operation, ShuffleNet can efficiently maintain high accuracy even while using limited computational resources with a robust generalization capability[7].

### F. Attention Mechanism

Attention mechanisms in deep learning are inspired by the cognitive attention process in humans. Just as humans focus on specific parts of their visual field to gather more information and ignore irrelevant parts, attention mechanisms in neural networks aim to identify and focus on the most informative parts of the input for better processing.

In the context of neural networks, especially those dealing with complex data such as images or sequential data (like text or time series), attention mechanisms can significantly enhance model performance. They do so by dynamically highlighting or giving more weight to certain features over others, allowing the model to concentrate on the most relevant aspects of the input data for the task at hand.

### G. Other Models for Testing Purposes

In addition to our primary five methods, we have developed several supplementary models for comparative analysis. Below, we provide an overview of their respective backgrounds:

1) *VGGNet* : The VGG-16 model, developed by Visual Graphics Group from Oxford [8], is a deep convolutional neural network known for its simplicity and depth. The model is characterised by its use of only 3x3 convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully connected layers, each with 4096 nodes, follow the convolutional architecture, and the model is finalised with a softmax classifier[8].

2) *DenseNet* : DenseNet, which refers to Dense Convolutional Network, is a type of convolutional neural network architecture that connects each layer to every other layer in a feed-forward fashion[9]. In traditional CNNs, the output of one layer is fed only to the next layer. In contrast, in DenseNet, each layer receives the feature maps of all preceding layers as input[9]. DenseNet is particularly notable for how it addresses the problem of efficiently training very deep neural networks by improving information flow and feature reuse.

## III. METHODS

### A. Data Augmentation

The exploration of data augmentation was motivated by the relatively small size of the ELPV dataset, which contained a total of 2,624 samples, as well as the pronounced class imbalance between the different defect probability labels.

With regards to the specific transformations applied to the training set, we experimented with horizontal and vertical flipping, rotation of up to 15 degrees, contrast enhancement and cutout. Cutout is a transformation that masks out a square region of fixed size at a random point in the image. Cutout has demonstrated its ability to improve the regularisation of CNNs on the CIFAR and SVHN datasets, and the transformation is motivated by reducing the overfitting of the network to the training set[10].

The transformations we chose to explore were appropriate in providing useful additional samples for the model to train on since it would be desirable for the model to achieve increased rotational invariance in detecting defective features. Since

certain features, such as thin cracks are subtle, transformations such as Gaussian blurring and random pepper noise were not chosen. To investigate its effectiveness in improving our model, the primary focus of the data augmentation technique was on the ResNet model.

### B. CNN(Conv2d)

CNN model with Conv2d(2-dimensional convolutional layer) is the first method we implemented in our project. It serves as a basic benchmark model for comparisons with other models. For this reason, we prioritized stability and simplicity as key features when designing this model. Before training the model, we resized the input images to 224, 224 and converted them to RGB, with 32 samples per batch. The model has two 2D convolutional layers, conv1 and conv2, designed for initial feature extraction. conv1 has 32 filters, and a kernel size of 3. conv2 has 128 input channels, and the same kernel size. Each convolutional layer is followed by a MaxPool2d pooling layer which is for reducing the spatial dimensions and helping in feature abstraction. Each pooling layer has a 2x2 window and a stride of 2, along with the ReLU activation function to reduce linearity. After pooling layers, we have the flattening layer to flatten the output from the convolutional layers to create a single long feature vector. Then a Dense layer with 512 units using ReLU activation is added to learn non-linearity. At the end of the layer sequence, there is an output dense layer with 4 units, using the softmax activation function for multi-class classification. The model is trained using Adam optimizer, with a learning rate of 0.0001. Along with the optimizer, the model uses Cross-Entropy Loss as the loss function. With the above sequential arrangement, this CNN model with Conv2d layer can properly serve the defectiveness classification task.

### C. ResNet

The ResNet model is the first model we built for transfer learning. For data preprocessing, the image dataset is converted to RGB and resized to 224x224 to align with the ResNet50 input requirements. The probabilities were categorized into 4 categorical label classes. After preprocessing, the input shape is set to (224, 224, 3). For the pre-trained model, we employed the ResNet50 model pre-trained on the ImageNet dataset as our base model. All layers of the base ResNet50 model were frozen to maintain the integrity of the pre-trained features. Then we added customized layers, which are a Global Average Pooling layer, a fully connected Dense layer with 1024 neurons with ReLU activation, and a final Dense layer with 4 neurons with softmax activation to adapt the pre-trained layers to our dataset. The model was compiled using the Adam optimizer with a learning rate of 0.0001 and the categorical cross-entropy loss function. After completing the model's coding, we ran 10 epochs using a batch size of 32 for training. To evaluate the impact of different numbers of layers in ResNet on the output results, we also trained ResNet101 and ResNet152 models with the same architecture and parameters for 10 epochs for comparative analysis.

#### D. ResNet Plus

In order to get to our "best" model for the purposes of this project, we developed enhancements to improve predictions on our data. In particular, we: Started with ResNet50, pre-trained on ImageNet as our base model. We experimented with unfreezing various layers but found the best results to be obtained with the base model layers fully frozen. We then incorporated the ability to utilise not only the image as features but additional categorical variables. This was done so as to be able to utilise the image type (mono vs. poly) as a feature in our model. The output from ResNet (pre-trained features) is concatenated with additional one-hot encoded features (image type). The concatenated output is then passed through a dense layer, ReLU activation, a dropout layer (to reduce overfitting), and fed through a final output layer of 4 output classes through softmax activation. The multi-input handling was a core feature of this model architecture, enabling two different types of input (image data and categorical data) to be utilised. This architecture would be scalable to incorporating additional categorical data, such as features extracted through unsupervised learning techniques. Furthermore, the above model architecture, together with data augmentation (through horizontal flipping) allowed us to get to our best model results (described as "ResNet Plus" in further results discussions).

#### E. ShuffleNet

Shufflenet is another pre trained convolutional neural network that we can utilize using the technique of transfer learning. A 2D convolutional layer takes in the 300x300 input image and creates a set of feature maps that are then passed through the shufflenet model. The model uses the SGD optimizer, a learning rate of 0.004, and a momentum of 0.9. It was trained in batches of 128.

Validation testing will be performed to determine the ideal parameters to be used in the first 2D convolutional layer as well as the number of epochs that results in the highest model accuracy. After this 3 models were trained. One on the mono images, one on the poly images and one on the combined dataset. Their accuracy was then assessed using the test set.

#### F. Attention Mechanism

In addressing solar panel damage detection, we utilised a custom Convolutional Neural Network (CNN) enhanced with Squeeze-and-Excitation (SE) blocks. This innovative model focuses on pivotal features within the images, thereby boosting its analytical capabilities.

1) *Model Architecture*: The model's framework is characterised by a sequence of convolutional layers, each succeeded by an SE block. These blocks are pivotal, recalibrating the channel-wise feature responses, thereby sharpening the model's focus on salient features indicative of panel damage.

2) *Convolutional Layers*: Initiated with a 2D Convolutional layer (32 filters, 3x3), the architecture progresses through similar layers, incrementing the filter count (64 and 128). These layers are integral for feature extraction.

3) *Squeeze-and-Excitation Blocks*: Each convolutional layer is coupled with an SE block, executing a global average pooling followed by dense layers. This setup enhances feature prioritisation.

4) *Pooling and Fully Connected Layers*: Post-convolutional processing, the network implements pooling and flattening, followed by a dense layer (128 neurons, ReLU activation), and a dropout layer to mitigate overfitting.

5) *Output Layer*: The culmination is a dense layer with four neurons (softmax activation), aligning with the damage categories.

6) *Training and Optimization*: Compiled with the Adam optimizer (learning rate: 0.001), the model underwent training over 10 epochs (batch size: 32). Their accuracy was then assessed using the test set.

#### G. Other Models for Testing Purposes

1) *VGG16* : In this project, the VGG-16 model pre-trained on ImageNet is utilised. The pre-training on a large dataset like ImageNet allows the model to establish an extensive range of feature detectors, from simple to complex, which can be beneficial for the task of image classification. For our specific application - identifying damage in solar panels - we adapt VGG-16 by removing its top layer (the fully connected layers and the output layer) and replacing it with layers designed for our classification task. The new top layer consists of a Flatten layer to convert the feature maps to a single vector, followed by a Dense layer with 256 units and ReLU activation for non-linear transformation. A Dropout layer with a 50% drop rate is added to reduce overfitting. Finally, a Dense output layer with four units and a softmax activation function is employed to classify the images into four categories based on damage level. The base layers of the VGG-16 model are frozen during training, which means their weights are not updated. This approach, known as transfer learning, leverages the pre-trained features of VGG-16 and fine-tunes the new top layers to adapt to our specific task. This method is particularly effective when working with small datasets, as it allows the model to use the complex features learned from a larger dataset. The model is compiled with the Adam optimizer, using a learning rate of 0.0001, and categorical cross-entropy as the loss function. The training is conducted over 5 epochs with a batch size of 32.

2) *Densenet* : A densenet model pre trained on the ImageNet data is used for this project. The first layer of densenet is replaced with a 2D convolutional layer which converts the input image to feature maps in a form that is then passed through the remainder of the densenet model. The version of densenet that is used is DenseNet121 which has 121 layers. It is the smallest of the DenseNet models available through pytorch and subsequently requires the least computational resources. A batch size of 32 was used along with a learning rate of 0.004 and a momentum of 0.9. The model was trained for 5 epochs.

## IV. EXPERIMENTAL RESULTS

The methods for this project are evaluated using a 4x4 confusion matrix and a classification report, which provide

a detailed view of performance across various classes. To obtain an unbiased evaluation, the dataset is divided into 75% for training purposes and 25% for testing. Accuracy, precision, and F1 scores will be our primary concerns. For further comparison of the effectiveness of our methods, after getting the confusion matrix and classification report for the whole dataset, we use two separate datasets, one with only monocrystalline cells and the other with solely polycrystalline cells, to check the effectiveness of our models for different types of solar panel cells. Separate confusion matrices were generated for each subset, helping us to make a comparative analysis to check if our methods showed a better performance or bias in identifying monocrystalline versus polycrystalline cells.

Figure 2 is the table displaying the accuracy scores for the five models we chose:

Result Table

	Conv2d	ResNet	ResNet Plus	ShuffleNet	ShuffleNet + Attention
Accuracy	0.70	0.74	0.79	0.76	0.58
Accuracy(mono)	0.72	0.74	0.78	0.79	0.55
Accuracy(poly)	0.78	0.74	0.80	0.74	0.59

Fig. 2. Result Table

#### A. CNN(Conv2d)

The performance of our CNN Conv2d model is summarised in Figure 3. According to this figure, it is notable that the conv2d model tends to misclassify across categories, especially for the ‘Possibly Defective’ category. The zero precision score indicates that this model doesn’t have the ability to correctly classify the ‘Possibly Defective’ category. The overall accuracy is 70%. Figure 4 and Figure 5 are the results for the monocrystalline cells-only and polycrystalline cells-only datasets. By comparing the results on these two matrices for different cell types, we can see that this CNN with Conv2d layers is more efficient in classifying polycrystalline cells.

#### B. ResNet

The performance of our ResNet model is summarized in Figure 6. The model has a high true positive rate for the ‘Fully Functional’ category, indicating that is highly effective in classifying the ‘Fully Functional’ category. However, it performs not that well in the ‘Likely Defective’ categories. The ‘Possibly Defective’ class had a precision of 0.20 but also a low recall score of 0.03, indicating that it has difficulty in accurately detecting this category. The ‘Likely Defective’ class had zero recall and precision, indicating it’s unable to identify this correctly. The overall accuracy is 74%. For the monocrystalline cells only and polycrystalline cells only datasets, the results are shown in Figure 7 and Figure 8. By comparing the results on these two matrices, we can see that

21/21 [=====] - 6s 296ms/step

Confusion Matrix:

```
[[ 99  0  1 79]
 [  4  0  1 21]
 [  8  0  6 60]
 [ 22  0  0 355]]
```

Classification Report:

	precision	recall	f1-score	support
Fully Functional	0.74	0.55	0.63	179
Possibly Defective	0.00	0.00	0.00	26
Likely Defective	0.75	0.08	0.15	74
Certainly Defective	0.69	0.94	0.80	377
accuracy			0.70	656
macro avg	0.55	0.39	0.39	656
weighted avg	0.68	0.70	0.65	656

Fig. 3. CNN General

Confusion Matrix:

```
[[ 53  0  1 25]
 [  2  0  1 11]
 [  3  0  0 26]
 [  7  0  0 140]]
```

Classification Report:

	precision	recall	f1-score	support
Fully Functional	0.82	0.67	0.74	79
Possibly Defective	0.00	0.00	0.00	14
Likely Defective	0.00	0.00	0.00	29
Certainly Defective	0.69	0.95	0.80	147
accuracy			0.72	269
macro avg	0.38	0.41	0.38	269
weighted avg	0.62	0.72	0.65	269

Fig. 4. CNN Mono

Confusion Matrix:

```
[[ 64  0  1 36]
 [  6  0  1  5]
 [  6  0 17 22]
 [  6  0  1 223]]
```

Classification Report:

	precision	recall	f1-score	support
Fully Functional	0.78	0.63	0.70	101
Possibly Defective	0.00	0.00	0.00	12
Likely Defective	0.85	0.38	0.52	45
Certainly Defective	0.78	0.97	0.86	230
accuracy			0.78	388
macro avg	0.60	0.50	0.52	388
weighted avg	0.76	0.78	0.76	388

Fig. 5. CNN Poly

ResNet is slightly more efficient in classifying monocrystalline cells.

Confusion Matrix for Entire Dataset:

```
[[361  2  0  9]
 [ 67  2  0  4]
 [ 24  1  0  6]
 [ 54  5  0 121]]
```

Classification Report for Entire Dataset:

	precision	recall	f1-score	support
Fully Functional	0.71	0.97	0.82	372
Possibly Defective	0.20	0.03	0.05	73
Likely Defective	0.00	0.00	0.00	31
Certainly Defective	0.86	0.67	0.76	180
accuracy			0.74	656
macro avg	0.44	0.42	0.41	656
weighted avg	0.66	0.74	0.68	656

Fig. 6. ResNet General

Confusion Matrix for Type mono:

```
[[142  0  0  5]
 [ 23  0  0  3]
 [ 14  0  0  4]
 [ 22  0  0 60]]
```

Classification Report for Type mono:

	precision	recall	f1-score	support
Fully Functional	0.71	0.97	0.82	147
Possibly Defective	0.00	0.00	0.00	26
Likely Defective	0.00	0.00	0.00	18
Certainly Defective	0.83	0.73	0.78	82
accuracy			0.74	273
macro avg	0.38	0.42	0.40	273
weighted avg	0.63	0.74	0.67	273

Fig. 7. ResNet Mono

Confusion Matrix for Type poly:

```
[[219  2  0  4]
 [ 44  2  0  1]
 [ 10  1  0  2]
 [ 32  5  0 61]]
```

Classification Report for Type poly:

	precision	recall	f1-score	support
Fully Functional	0.72	0.97	0.83	225
Possibly Defective	0.20	0.04	0.07	47
Likely Defective	0.00	0.00	0.00	13
Certainly Defective	0.90	0.62	0.73	98
accuracy			0.74	383
macro avg	0.45	0.41	0.41	383
weighted avg	0.68	0.74	0.68	383

Fig. 8. ResNet Poly

To assess the influence of different layer depths in ResNet on model accuracy, we employed three different variants of

ResNet: ResNet50, ResNet101, and ResNet152. Figure 9 are the line charts for comparison.

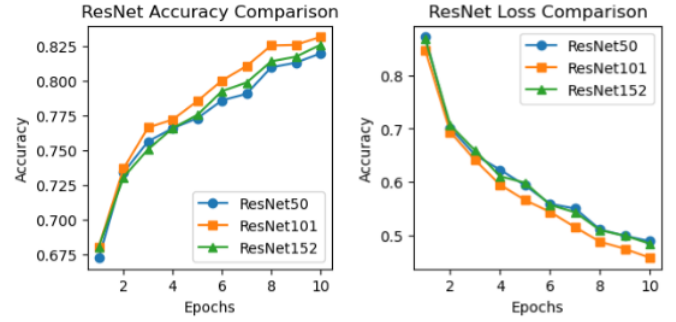


Fig. 9. Results of ResNets with Different Layers

### C. ResNet Plus

ResNet Plus has the best overall results with 79% accuracy. Figure 10 is the summarization. Compared to the original ResNet model, it is evident that the overall accuracy for each class is improved. The original ResNet model was unable to correctly classify the “Likely Defective” category while the ResNet Plus model has a limited ability to correctly classify. According to the report results for poly and mono types in figure 11 and 12, we can see that ResNet Plus is more effective when classifying for polycrystalline cells dataset.

Confusion Matrix for Entire Dataset:

```
[[335 23  0 14]
 [ 25 41  1  6]
 [ 17  4  3  7]
 [ 27  8  5 140]]
```

Classification Report for Entire Dataset:

	precision	recall	f1-score	support
Fully Functional	0.83	0.90	0.86	372
Possibly Defective	0.54	0.56	0.55	73
Likely Defective	0.33	0.10	0.15	31
Certainly Defective	0.84	0.78	0.81	180
accuracy			0.79	656
macro avg	0.64	0.58	0.59	656
weighted avg	0.78	0.79	0.78	656

Fig. 10. ResNet Plus General

### D. ShuffleNet

Shufflenet has multiple parameters that need to be optimized if the model is to perform well. To perform hyperparameter tuning various values for the parameters were used in the creation of the model which was then trained and assessed using the validation data. The accuracy line chart is shown in figure 13. The first optimum number of epochs was determined by comparing the validation accuracy produced by each number of epochs from 1 to 100. For the first 40 epochs there seems to be a trend of increased accuracy with more epochs but after

21/21 [=====] - 29s 1s/step

Confusion Matrix for Type mono:

```
[[137  9  0  1]
 [ 9 13  1  3]
 [ 10  2  3  3]
 [ 12  5  4 61]]
```

Classification Report for Type mono:

	precision	recall	f1-score	support
Fully Functional	0.82	0.93	0.87	147
Possibly Defective	0.45	0.50	0.47	26
Likely Defective	0.38	0.17	0.23	18
Certainly Defective	0.90	0.74	0.81	82
accuracy			0.78	273
macro avg	0.63	0.59	0.60	273
weighted avg	0.78	0.78	0.77	273

Fig. 11. ResNet Plus Mono

Confusion Matrix for Type poly:

```
[[198 14  0 13]
 [ 16 28  0  3]
 [  7  2  0  4]
 [ 15  3  1 79]]
```

Classification Report for Type poly:

	precision	recall	f1-score	support
Fully Functional	0.84	0.88	0.86	225
Possibly Defective	0.60	0.60	0.60	47
Likely Defective	0.00	0.00	0.00	13
Certainly Defective	0.80	0.81	0.80	98
accuracy			0.80	383
macro avg	0.56	0.57	0.56	383
weighted avg	0.77	0.80	0.78	383

Fig. 12. ResNet Plus Poly

45 the accuracy does not significantly trend in any direction. 45 was chosen as it produces amongst the highest accuracy while also being small enough to not significantly slow down computation speed. Using 45 epochs, validation testing was performed for various values of kernel size, stride, padding and bias inclusion.

The shufflenet model was trained and evaluated on the test set and this was the final train and test accuracy results in Figure 14. And mono and poly classes results are produced in Figure 15 and Figure 16.

### E. ShuffleNet with Attention Mechanism

When shufflenet was used with attention Figure 17 is the test accuracy results and confusion matrix results for the entire dataset. And Figure 18 and Figure 19 are the results for mono and poly datasets.

### F. Other Models for Testing Purposes

Figure 20 is the result table for all other models we implemented for comparison.

Validation Accuracy vs Epochs for shufflenet

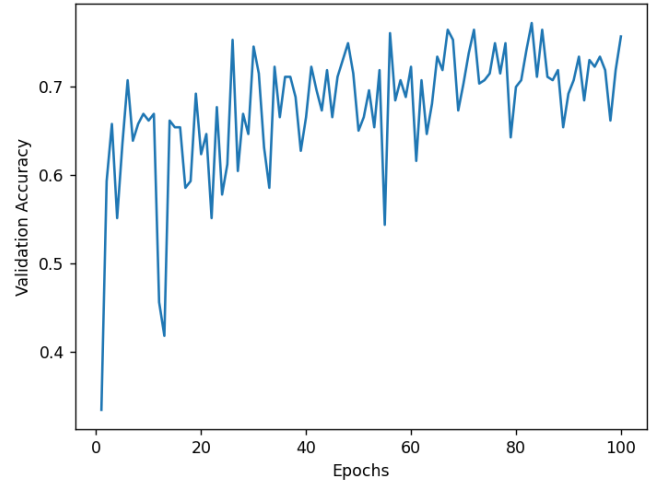


Fig. 13. ShuffleNet Val Accuracy

```
Epoch 45, Loss: 0.027211826075525844, Acc: 0.9947594092424964
Test Loss: 1.0364096810420353, Test Acc: 0.7557251908396947
Precision: 0.6509562841530054, Recall: 0.5342519105999739, F1: 0.5703130469435783
```

		0	0.33	0.66	1
True	0	138.0	3.0	3.0	7.0
	0.33	15.0	11.0	0.0	3.0
	0.66	5.0	0.0	2.0	4.0
	1	22.0	1.0	1.0	47.0

Fig. 14. ShuffleNet General

```
Epoch 45, Loss: 0.046956578003508706, Acc: 0.9848661233993015
Test Loss: 1.0544097423553467, Test Acc: 0.794392523364486
Precision: 0.7192128438268428, Recall: 0.6150550060473516, F1: 0.643452380952381
```

		0	0.33	0.66	1
True	0	54.0	1.0	0.0	4.0
	0.33	3.0	7.0	0.0	1.0
	0.66	4.0	0.0	1.0	1.0
	1	6.0	1.0	1.0	23.0

Fig. 15. ShuffleNet Mono

```
Epoch 45, Loss: 0.005510459537617862, Acc: 1.0
Test Loss: 1.6069799065589905, Test Acc: 0.7354838709677419
Precision: 0.5102383053839364, Recall: 0.47983091787439613, F1: 0.49183510553373566
```

		0	0.33	0.66	1
True	0	81.0	5.0	0.0	6.0
	0.33	8.0	7.0	2.0	1.0
	0.66	3.0	2.0	0.0	0.0
	1	11.0	1.0	2.0	26.0

Fig. 16. ShuffleNet Poly

```
Epoch 45, Loss: 1.0545333623886108, Acc: 0.5745593139590282
Test Loss: 1.0574114720026653, Test Acc: 0.5763358778625954
Precision: 0.14408396946564886, Recall: 0.25, F1: 0.1828087167070218
```

		0	0.33	0.66	1
True	0	151.0	0.0	0.0	0.0
	0.33	29.0	0.0	0.0	0.0
	0.66	11.0	0.0	0.0	0.0
	1	71.0	0.0	0.0	0.0

Fig. 17. ShuffleNet with Attention General

```

Epoch 45, Loss: 1.0859943628311157, Acc: 0.5471478463329453
Test Loss: 1.0834612846374512, Test Acc: 0.5514018691588785
Precision: 0.1378504672897196, Recall: 0.25, F1: 0.17771084337349397
Predicted
    0  0.33  0.66  1
True
0    59.0  0.0  0.0  0.0
0.33 11.0  0.0  0.0  0.0
0.66  6.0  0.0  0.0  0.0
1    31.0  0.0  0.0  0.0

```

Fig. 18. ShuffleNet with Attention Mono

```

Epoch 45, Loss: 1.02250075340271, Acc: 0.5935483870967742
Test Loss: 1.0739555656909943, Test Acc: 0.5935483870967742
Precision: 0.14838709677419354, Recall: 0.25, F1: 0.18623481781376516
Predicted
    0  0.33  0.66  1
True
0    92.0  0.0  0.0  0.0
0.33 18.0  0.0  0.0  0.0
0.66  5.0  0.0  0.0  0.0
1    40.0  0.0  0.0  0.0

```

Fig. 19. ShuffleNet with Attention Poly

Result Table (Testing Models)

	VGG16	VGG16 + Attention	CNN + Attention	Conv2D with Different Layers	DenseNet
Accuracy	0.75	0.64	0.69	0.62	0.73

Fig. 20. Testing Models Accuracy Results

## V. DISCUSSION

After meticulously inspecting the results of the models we built, we can draw the conclusion that ResNet Plus has the best performance with a general accuracy of 79%, while ShuffleNet and ResNet are slightly slow behind with 76% and 74% accuracy. On the other hand, ShuffleNet with Attention Mechanism falls behind with the lowest accuracy in our evaluations. Also, when including those models for testing considerations, we can find that VGG16 holds a competitive accuracy which is as same as ResNet Plus's accuracy. This result approves the effectiveness of the improvement and data augmentation we did to the ResNet model.

However, all of our models have a low accuracy when classifying for the "Likely Defective" and "Possible Defective" classes. We assume that this inability is caused by the poor performance of our data augmentation and the similarity of the images in these two categories. In the future, better data augmentation techniques can be deployed to improve the accuracy of classifying these two categories. In Figure 21, there are some examples that all our models succeeded and failed to predict. We can see that our models are performing terribly, especially when there's a blur mark on the input image. They always classify them as fully functional while possibly being defective. After discussing about this kind of error, we think we will need better image segmentation techniques to improve the effectiveness of our methods in the future. In the evaluation of the performance to classify the defectiveness

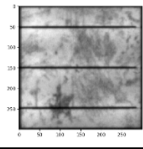
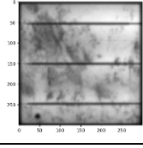
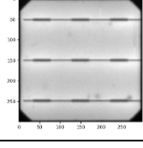
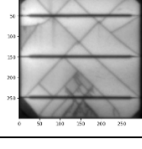
Data Sample	True Label	Prediction	Result
	Fully Functional	Fully Functional	Succeed
	Possibly Defective	Fully Functional	Failed
	Likely Defective	Fully Functional	Failed
	Certainly Defective	Certainly Defective	Succeed

Fig. 21. Prediction Samples

for monocrystalline and polycrystalline cells, it's evident that Shufflenet has the best performance in classifying defects in monocrystalline cells with 79% accuracy, and ResNet Plus has the best performance in classifying defects in polycrystalline cells with 80% accuracy.

For the ResNet models with different layers, according to the chart in Figure 9, it's obvious that ResNet101 outperformed the other models in our project with the highest accuracy and the lowest loss rate. This outcome was surprisingly different from our initial expectation, which was that ResNet models with more layers would yield higher accuracy. This teaches us a lesson that it is better to pick the ResNet variant that is best suited to our tasks, rather than assuming that increased complexity will always lead to better performance.

In comparing our research work to the original Research Paper, We were not able to achieve the accuracy of the original research papers by S. Deitsch et al and A. Korovin et al. For their paper, they have more extensive studies leveraged a variety of methods including key point and feature extraction, unsupervised anomaly detection, thresholding, and deep learning. However, we were able to improve on the standard ResNet model by incorporating additional features and improving the scalability and optimization (e.g. unsupervised learning). In the future, we plan to explore and implement more advanced data preprocessing techniques and modeling approaches to our models to match the accuracy level of the original research paper.

## VI. CONCLUSION

In conclusion, our project successfully utilized multiple methods to fulfill the tasks of the detection and classification



of defects in solar panels using electroluminescence (EL) imagery. Among all the models we built for testing, our ResNet Plus model holds the highest general accuracy affirming its effectiveness in classifying solar cell tasks. However, this model, along with our other developed models, still has a limitation in that it has a weak ability to classify the “Likely Defectiveness” category with 0.67 probability and the “Possibly Defectiveness” category with 0.33 probability.

Our research also faces other limitations, including a tight timeframe and limited computational resources. Initially, we developed ten models, but due to time constraints, we could only test five of them, which are our 5 primary results now. We also have limited GPU and memory power, which made our training and testing time longer. In the future, we will improve our research by further refining the model architectures, exploring advanced optimization techniques, and expanding the dataset through feasible methods such as transformation.

## REFERENCES

- [1] Buerhop-Lutz, C.; Deitsch, S.; Maier, A.; Gallwitz, F.; Berger, S.; Doll, B.; Hauch, J.; Camus, C. & Brabec, C. J. A Benchmark for Visual Identification of Defective Solar Cells in Electroluminescence Imagery. European PV Solar Energy Conference and Exhibition (EU PVSEC), 2018. DOI: 10.4229/35thEUPVSEC20182018-5CV.3.15
- [2] L. Perez and J. Wang, “The Effectiveness of Data Augmentation in Image Classification using Deep Learning.” arXiv, 2017. doi: 10.48550/ARXIV.1712.04621.
- [3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). “Deep learning.” *Nature*, 521(7553), 436-444.
- [4] DesiCrew Solutions Private Limited. (2023). “Leveraging Transfer Learning for Computer Vision.” LinkedIn[Online]. Available: <https://www.linkedin.com/pulse/leveraging-transfer-learning-computer-vision-desicrew-solutions/>
- [5] He, K., Zhang, X., Ren, S., & Sun, J. (2015). “Deep Residual Learning for Image Recognition.” arXiv preprint arXiv:1512.03385.
- [6] Qingyuan Wang, Antoine Frappé, Benoit Larras, Barry Cardiff, Deepu John. Complexity Reduction of CNNs using Multi-Scale Group Convolution for IoT Edge Sensors. 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Oct 2022, Glasgow, United Kingdom. pp.1-4, ff10.1109/ICECS202256217.2022.9970790ff. fhal-03938518
- [7] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” arXiv, 2017. doi: 10.48550/ARXIV.1707.01083.
- [8] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv, 2014. doi: 10.48550/ARXIV.1409.1556.
- [9] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017.
- [10] T. DeVries and G. W. Taylor, “Improved Regularization of Convolutional Neural Networks with Cutout.” arXiv, 2017. doi: 10.48550/ARXIV.1708.04552.