# ELPV dataset classification

**(GroupAlloc9)**

Mo Li z5442761 Weilin Sun z5457516 YenJung Huang z5437775

Ziang Zhu z5436962 Shuo Cheng z5444192

## I. Introduction

In today's world, as humankind's demand for sustainable energy increases, solar energy, as a clean and renewable energy source, is becoming more and more important in society. Solar panels, especially those installed in remote areas or for mobile devices, provide us with a cost-effective and environmentally friendly power solution. However, these photovoltaic (PV) cells face the risk of damage caused by environmental factors such as wind, rain and hail, as well as manufacturing defects. These damages can not only affect the panel's efficiency but may also reduce the panel's service life. Therefore, in order to maintain the optimal operating performance of the panels, it is very important that we monitor the health of the PV cells. This project is to develop a method using electroluminescence (EL) imaging and computer vision technology that can adopt an efficient and accurate prediction and classification method, so that it can find potential defects in solar panels, thereby ensuring the reliability of solar systems. and high efficiency.

From our team's own perspective, this project was not only a technical challenge, but also about delivering innovation and increasing precision. In this task, what we want to do is how to accurately classify and predict defects on solar panels. This requires us to use advanced computer vision technology to analyze and interpret the health of PV cells by combining it with EL imaging. This way we can quickly identify batteries whose functionality is already causing problems, allowing us to take early steps to prevent a loss of efficiency. In addition, the process should be able to identify which panels have defects that do not affect their performance, thus avoiding unnecessary repairs or replacements.

The ELPV dataset used in the project contains 2,624 EL images from 44 different solar modules, representing photovoltaic cells in different health states. Each image is labeled with defect probability (0-1) and solar module type (monocrystalline or polycrystalline). These images are normalized to ensure consistency and accuracy of analysis. For example, we can see that the image "cell1.png" shows a smooth polycrystalline silicon cell surface, while "cell2.png" shows a monocrystalline silicon cell with obvious defects. With these samples, I think we can gain insights into the characteristics of different types and health conditions of photovoltaic cells.

Although the ELPV dataset provides an exhaustive basis for analysis, the project itself presented several challenges. First, we need to develop an algorithm that can accurately identify and classify tiny defects, which we believe is a major challenge for the field of computer vision. Second, because the health of a PV cell is revealed by subtle changes in its surface, we need to use methods with extremely high resolution and sensitivity. In addition, considering that practical applications of solar panels may encounter extreme weather, our solutions also need to be able to adapt to different lighting conditions and environmental changes. These challenges require us to continue to innovate in technology development and application to ensure the effectiveness and practicality of methods.

## II. Literature Review

Defect detection for EL images is active topic in recent years. There are several different methods proposed to solve this problem.

### *A. Traditional Machine Learning*

Traditional machine learning method is used to solve this problem. Generally, it can be described as following steps: pre-processing image for better feature extraction, feature extraction and selection, and then use a classifier to classify defect and non-defect. Sergiu [1] follows the tradition method, tested with different feature extraction method, and use SVM and CNN as classifier. Another very interesting approach proposed by Tsai [2] uses ICA to extract the basis image, and then calculate similarity between testing image and basis image. This method can only detect defect/non-defect but cannot find percentage defect. I also included this method in the code, so I can give a taste of different approaches.

When dealing with datasets, class-imbalanced problem is significant. SVM on the other hand is very sensitive to noises. Chapter 5 of He [3] gives a very good overview of how to deal with class-imbalanced problem. Imbalance problem in SVM causing decision boundary biased toward the majority class, e.g., more testing sample will be classified as majority class. To solve this problem, this book [3] provided several methods

such as, uses different sampling method (up sampling, and down sampling), apply class ratio to parameter C (Different error cost), that is use different C to different classes. Rehan Akbani [4] also proposed several methods to solve this problem, such as use SMOTE method for up sampling, and use different error cost to enhance the performance. Other methods are also considered, for example, kernel modification methods. This method uses class boundary alignment (CBA) to enlarge more of the class boundary around the minority class, that is, the distance of "surrounded area" are increased for minority class, that will push the decision boundary towards majority class.

Features extraction is also very important for this problem. In this project, I examined several different features extraction method. SIFT, SURF, ORB and KAZE are most common descriptors used in computer vision. Shaharyar [5] compared all methods mentioned above in different aspects, and result shows that SIFT are found to be most accurate algorithm on average. Sergiu [1] further compared SIFT and SURF, and the result shows SIFT perform well on this dataset.

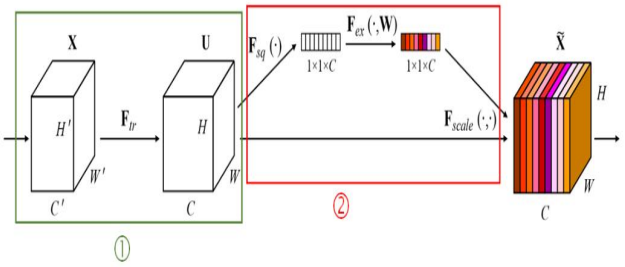
Once the feature descriptors are obtained, next step is to form feature vectors. One can simply uses descriptor as feature vector by concatenate all descriptor together, however, the constraints would be different images can generate different amount of feature vectors. Most common approach would be using bag of virtual words. Gabriella's [6] Journal article provide comprehensive conclusion using this method, and in this paper SIFT method was also used. Other methods to form feature vectors include Feature Pyramid [7], Spatial Pyramid Matching [8] are considered but not used in this project.

## *B. Deep Learning*

### *1) Mobile Net V3*

MobileNetV3 is a highly efficient neural network architecture optimized for mobile and edge devices, featuring lightweight attention mechanisms, hardware-aware network search, and advanced activation functions. [9]

Features of MobileNetV3:
Inverted Residual:
It uses a lightweight depth-wise separable convolution followed by a linear bottleneck.
This design enables the network to learn fine-grained features efficiently, reducing computational cost while maintaining performance.
Linear Bottleneck:
As a part of Inverted Residual, it involves linearly transforming the output of the depth-wise separable convolution before it is passed to the next layer. This reduces the dimensionality of the data, leading to efficiency.
Squeeze-and-Excitation Module:
The Squeeze-and-Excitation (SE) block is a form of attention mechanism in neural networks. It adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. This helps the network focus more on informative features.[10]

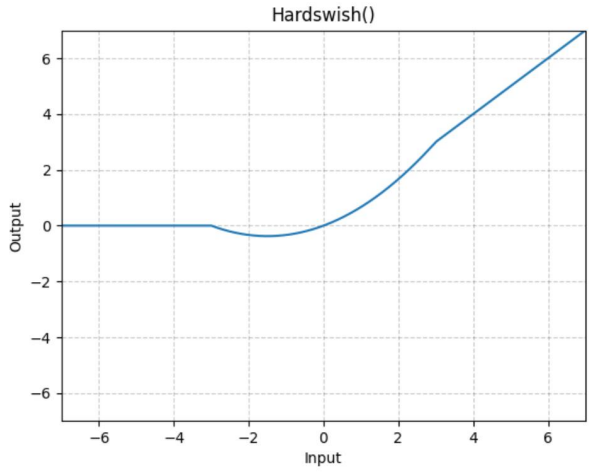



Global Average Pooling:
Comparing with the fully connected layer, GAP has better anti-exportability due to the parameter population of the fully connected layer. The global average pooling layer does not require parameters to avoid overfitting in this layer. Global average pooling sums spatial information and is more robust to spatial changes in input.[11]

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j)$$

Hardswish activation function:
It uses hardswish for activation functions that is fast in computing and friendly to quantitative process, which enhances network activation efficiency.[9]

$$\text{Hardswish}(x) = \begin{cases} 0 & \text{if } x \le -3, \\ x & \text{if } x \ge +3, \\ x \cdot (x+3)/6 & \text{otherwise} \end{cases}$$




### *2) SMOTE*

SMOTE is a popular synthetic minority class oversampling technique used to deal with imbalanced data set problems in machine learning. It improves the performance of the

classifier by synthesizing new examples in the feature space to increase the number of samples in the minority class[13].

In the project, we use smote to synthesize new minority class samples to balance the data set and improve the model's performance in detecting rare defects. You can also increase the sample size of the minority class to improve the generalization ability of the model, and you can also use SMOTE to balance the class distribution, thereby reducing the risk of overfitting.

## III. METHODS

In this section, we will discuss different methods used for this project. Two machine learning methods are proposed, ICA method and SVM classifier.

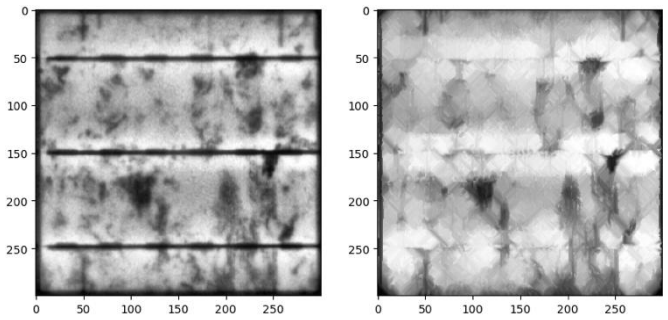
### *A. ICA methods*

The first method I use is following Tsai's reconstruction method [2].

#### *1) Data Preprocessing*

From some basis data analysis, EL image has randomly shaped dark regions in the background, these regions are not considered as defect. Defects are dark line shape or bar shape regions. The purpose of preprocessing is these randomly shaped region can be removed, meanwhile, retain the line-shape or bar-shape defects.

Tsai proposed to use morphological smoothing. Morphological are used for contrast-enhancement, texture description, edge detection and thresholding. We only use morphological smoothing; it contains two basic method erosion and dilation. Erosion assigns each pixel the minimum value found over the neighborhood of the structuring element, and dilation does exactly the opposite. Dilation will assign each pixel the maximum value found in structuring elements. In this project, three structuring elements are designed and compared. Since I want to remove the random noise in the background but keep the crack line or bar, the length of structuring elements should longer than most random shaped regions but shorter than the cracks. This number is proposed as 13 in original paper, but the image in original paper is shaped 206x206. Our project has image size of 300 x 300, so I linearly increase the length. Empirically, this number is given as 17 produces optimal results. Although cracks can be in all directions, but it's not possible to cover all directions. Three directions are used 45-degree, 135 degree and 90 degrees. They are two diagonals in 17x17 matrix, and vertical vector. 0 degree is purposely not used. Since part of image contains 2 horizontal lines, but not others, it can be easily considered as cracks, so these two lines should be removed. If 0-degree structure elements are preserved, then it will be still as it is in the image. Below image shows effect after morph smoothing.




For each pixel in the image, we select the SE with minimum accumulated gray levels (sum of intensity within each SEs). The smallest direction will be selected as morphological dilation smoothing. Quote from original paper: "If -SE contains all defect points, the dilated value will be still small for a dark region. Otherwise, the dilated value will be large for a bright region in the background [2]." Then, dilation will be used to select the highest value in the SE, and replace original pixel. This preprocessing method is very efficiency to remove the background noise and retain the defect.

#### *2) Image reconstruction*

Tsai believe that the test image can be reconstructed by linear combination of basis images. This idea came from ICA decomposition. ICA can separate noise from observed signal, $U = W \cdot X$ where $X$ is observed signal, $W$ is the de-mixing matrix, and $U$ is the estimation of source signal. $W$ maximize the negentropy for each independent component. If the images are "perfect" images plus random noise, then we can derive these random noise from basis images. Here, we make assumption that "random shape dark region" are independent and random. Once we obtain the original "perfect" image (basis image) without any random shade, then we can either use this to extract feature vector or reconstruct the image by representing test image as a linear combination of the learned basis image. Original paper tried both method, reconstruction method has better accuracy, so we only use reconstruction method.

Each test image can be represented by a coefficient vector and basis image, eg: $\hat{y} = b \cdot U$, b can be found by $b = y \cdot U^{+}$, where $U^{+}$ is pseudo-inverse of U. Coefficient vector can also use as feature vector for other classifier. Reconstruction method uses this coefficient vector to reconstruct test image y. Once images are reconstructed, simply calculate the different between reconstructed image and test image, If the test sample contains defects, then it is expected that the error should be large. Below is calculation for the error.

$$\Delta\,\epsilon(y) = \left|\left|y - c \cdot \hat{y}\right|\right|$$

$$c = \frac{\left|\left|y\right|\right|}{\left|\left|\hat{y}\right|\right|}$$

Here c is constraint parameter used as regularization. When this error is greater than a predefined threshold, then it considered as defect otherwise its non-defect. I further extend this idea to set three different thresholds, and each one corresponding to a different percentage of defects.

Tsai also proposed an easier way to calculate $\hat{y}$ as below:

$$\begin{aligned}\hat{y} &= b \cdot U \\ &= (y \cdot U^{+})(WX) \\ &= y(WX)^{T}[(WX) \cdot (WX)^{T}]^{-1}(WX) \\ &= y \cdot X^{T}[X \cdot X^{T}]^{-1}X \\ &= (y \cdot X^{+}) \cdot X\end{aligned}$$

Use this way, we don't need to calculate ICA, and can simply use original matrix X calculate reconstructed test image.
Threshold are calculated as follow. We first select all defect-free data in train set and we split this subset to 5-folds. We use 4 folds as X, and remaining fold as y, since all images are defect free, we can calculate the mean error and standard deviation of this error. We take mean of mean error from 5 folds and mean standard deviation as error standard deviation. The threshold is then mean error plus one, two and three standard deviations for each 33%, 66% and 100 % defect threshold.

We can use same method on test images and compare the error with threshold values.

### *B. SVM method*

Second method we use is feature extractions and SVM classifier.

#### *1) Augumentations*

Purpose of augmentation is to increase variety of sample and increase model's ability to generalization. Several methods were considered in this project, include random noises, rotation, and flip. Here rotation and flip are only applied to method that is variant to rotations, and flip. SIFT on the other hand produce same feature vector use rotation and flip.

#### *2) Preprocessing*

Image preprocessing can remove noises and can help feature extraction method to extract more accurate feature. I have tried a several methods: CLACHE contract, Gaussian blur prior to Laplacian for edge detection, local binary pattern, morphological opening, and morphological smoothing. Since there are no standard for what a good feature after preprocessing is, I can only try each combination. Each combination is tested, and only the preprocessing method yields best result will be used for feature extractions.

For parameters, I must visualize its effect to determine which one "looks like" gives better result, however this method is not rigorous.

#### *3) Feature extraction*

Only SIFT and HOG descriptors are used in this project.
SIFT descriptor can be extracted from majority of images, however, for some images that is too dark, sift unable to extract any descriptors.

We later built a bag of visual words on extracted feature descriptor. All descriptors are first using different K value to find clusters. Each cluster represent how many different unique descriptors we can find in the image. Since we have 8 different classes, we can make assumption that each class can be represent by 10 different clusters of descriptors. For example, line-shaped scare can be detected as a descriptor, and amount of line shaped crack can then use as feature vector in classifiers. K value is set from 80, 160, 320, and 640. That is, we assume that there are n * 10 unique features in each image. Then we run Kmean algorithm to clustering all descriptors to K clausters, if it close to any clusters, the corresponding bin will increase one. Some images with 0 descriptors will obtain all 0 in the histogram. The resulting feature is length of K for each image. Then we can use this histogram run classification use SVM

HOG feature also considered in this project. HOG is efficient in object detection task; it can capture the appearance and shape of an object in image by finding distribution of intensity gradients and edge directions. HOG run as following steps: it first creates small cells, in our case 20x20 cell in the image, this is because majority of patterns in the image is within 30 pixels. For each cell, it will compute a histogram of gradient directions to capture local features. For global features, it will further combine these cells to blocks. HOG unlike SIFT, it will combine all these descriptors into bins, number of bins are set to be 10 (HOG original proposed 9 bins, but I empirically tested that 10 would be better number bins). A window will be slide on entire image, and each window would be a block. then a feature vector is obtained for each sliding. The number of cells per block are sets to 5, again this is we tested with different number, and we find that 5 gives better classification result.

From this settings we end up with a feature vector length 30250, this would be too large even for SVM to handle. We further apply principle component analysis to reduce the dimensions. We kept only 85% of information, and resulting vector length is 99. We further normalize this feature vector and throw it into model.

#### *4) Model selections*

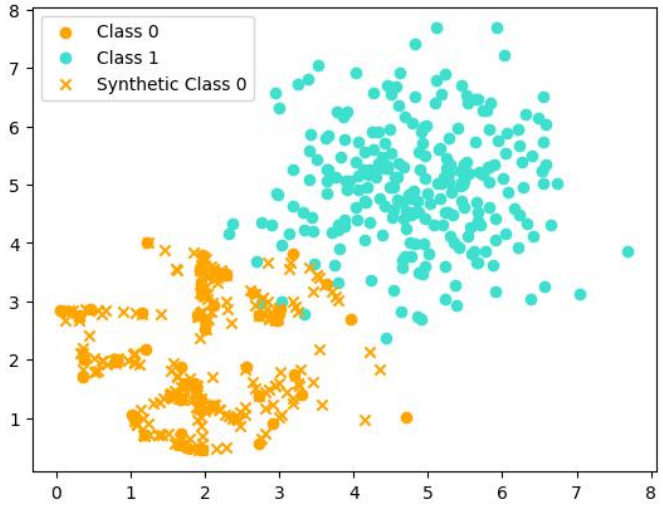
Different classification model was considered. I cross-validated KNN, SVM, Forest, Regression, ada-boost, and gradient boost. Comparing each of its result, I found that SVM would be best in this project. Cross-validation on models is

only used default parameters without fine-tuning. After I have decided to use SVM, I rigidly test different parameter for SVM.
As mentioned in earlier sections, there are lots of application using SVM for EL images, and in this specific dataset, class-imbalance is causing a big issue.

SMOTE method is used along with DEC method to fix imbalanced class. SMOTE find the training data x and its closet neighbor z under same class. A new data u will be generated between x and z.

$$u = x + w * (z - x)$$

Where w is a random number between 0 to 1. Using 2-D example,




Above figure demonstrated synthetic data (x mark) are generated for class 0.

Several other technique was also along with SMOTE and DEC. Down sampling proven to be worst among them all, up sampling with smote on one vs one model, that is train 3 + 2 + 1 models for all pairs of classes, and use vote to decide final output. Another method I have tried is one vs other method, that is train a classifier for each class, in our case, 4 classes.

### *C. ResNet Method*

ResNet, or Residual Network (ResNet), was introduced in 2015 by researchers at Microsoft Research. It represents a major innovation in the field of deep learning, specifically addressing the problem of network degradation during deep neural network training. As networks become deeper, training difficulty and performance tend to decline - a phenomenon known as "degradation." The concept of "residual learning" introduced by "ResNet" aims to alleviate this problem by allowing certain layers to learn the residuals (differences) between inputs and outputs, rather than mapping them directly.

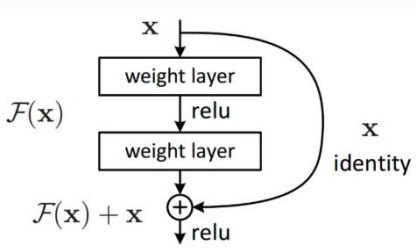



Figure 1 Residual learning: a building block

There are many variants of ResNet, and each member of this family is designed to solve common problems in deep neural networks such as vanishing gradients. The core innovation of ResNet is its "residual blocks", which allow the network to pass information directly through jump connections (or layer-hopping connections), thus effectively training deeper ResNet. The original and best-known members of the ResNet family are ResNet-18 and ResNet-34, which are relatively "shallow" models with 18 and 34 layers, respectively, and are suited for computationally constrained environments and for tasks that do not involve particularly large amounts of data. Despite their small number of layers, they still inherit ResNet's core strengths, such as mitigating the problem of vanishing gradients and improving the efficiency and accuracy of training deep networks. These models perform well in tasks such as image recognition, especially in scenarios that do not require extreme deep network processing. And our task this time is to process 2624 300*300 images, which is obviously a small sample size problem. So we chose ResNet18 as the model to solve the problem.

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer | 152-layer |
|---|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | | |
| | | 3×3 max pool, stride 2 | | | | |
| conv2_x | 56×56 | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ | $11.3\times10^9$ |

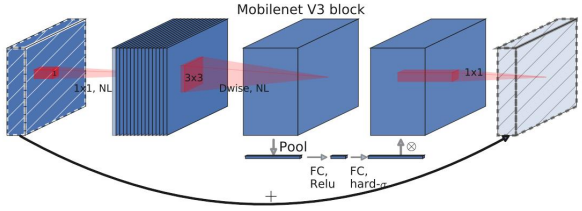
Figure 2 Family members of ResNet

ResNet-18, as its name suggests, consists of 18 layers, including convolutional, batch normalization, activation layer, batch normalization layer, activation layer, and fully connected layer. The clever structure of these layers improves learning efficiency and accuracy.
In this project we implemented ResNet-18 model using PyTorch framework. During training, we used cross-entropy loss function and Adam optimizer. The training and test sets are 75% and 25% respectively. To address the problem of category imbalance, we used the oversampling technique (SMOTE) to increase the number of samples in some categories.
To train the model, all images are converted into tensor and normalized. Normalization is done by subtracting the mean of the image and dividing it by its standard deviation, which

helps to speed up the convergence of the model training and improve its performance. In addition, during the experiments we found that the data type with 8 classifications (mono and poly are divided into different categories) is better than 4 classifications, both in terms of accuracy and recall. That is, the same model with the same parameters and learning rate will always outperform the 4-categorized data type by 2% to 3% when using the 8-categorized data type.

### D. MobileNetV3 Method

Mobile Net V3 is a lightweight deep learning architecture optimized for mobile devices, and it is the latest addition following the V1 and V2 versions. Mobile Net V3 is designed to take advantage of hardware-aware network architecture search and the NetAdapt algorithm, which optimizes the accuracy and latency balance[12].




MobileNetV3 has efficient building blocks that can effectively decompose traditional convolutions into lightweight depth convolutions and heavier 1x1 point convolutions, and can also take advantage of the low-rank nature of the problem to build more efficient layer structures. MobileNetV3 is also very good at network search and improvement. It leverages platform-aware NAS to search the global network structure and optimize each network block. Combined with the NetAdapt algorithm, the number of filters is searched layer by layer to find the optimized model for the given hardware platform. In addition, MobileNetV3 also introduces a new nonlinear activation function h-swish, making its calculation faster and more suitable for quantization[12].
MobileNetV3 Large Specification:

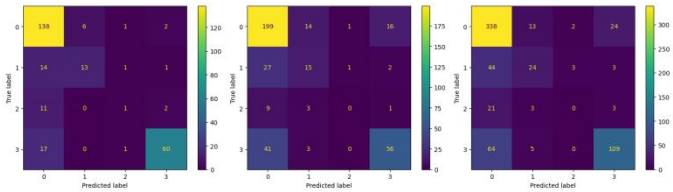
| Input | Operator | exp size | $\#out$ | SE | NL | $s$ |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 16 | - | HS | 2 |
| $112^2 \times 16$ | bneck, 3x3 | 16 | 16 | - | RE | 1 |
| $112^2 \times 16$ | bneck, 3x3 | 64 | 24 | - | RE | 2 |
| $56^2 \times 24$ | bneck, 3x3 | 72 | 24 | - | RE | 1 |
| $56^2 \times 24$ | bneck, 5x5 | 72 | 40 | ✓ | RE | 2 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 5x5 | 120 | 40 | ✓ | RE | 1 |
| $28^2 \times 40$ | bneck, 3x3 | 240 | 80 | - | HS | 2 |
| $14^2 \times 80$ | bneck, 3x3 | 200 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 184 | 80 | - | HS | 1 |
| $14^2 \times 80$ | bneck, 3x3 | 480 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 3x3 | 672 | 112 | ✓ | HS | 1 |
| $14^2 \times 112$ | bneck, 5x5 | 672 | 160 | ✓ | HS | 2 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | bneck, 5x5 | 960 | 160 | ✓ | HS | 1 |
| $7^2 \times 160$ | conv2d, 1x1 | - | 960 | - | HS | 1 |
| $7^2 \times 960$ | pool, 7x7 | - | - | - | - | 1 |
| $1^2 \times 960$ | conv2d 1x1, NBN | - | 1280 | - | HS | 1 |
| $1^2 \times 1280$ | conv2d 1x1, NBN | - | k | - | - | 1 |

## IV. Experimental Results

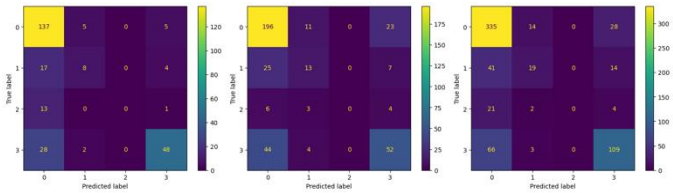
There is total 656 data in the dataset, and 377 data (0.57) are 0 percent defect. 178 out of 656 are class 100 defect (0.27). So, the base line is 57%.

### A. Machine learning Method

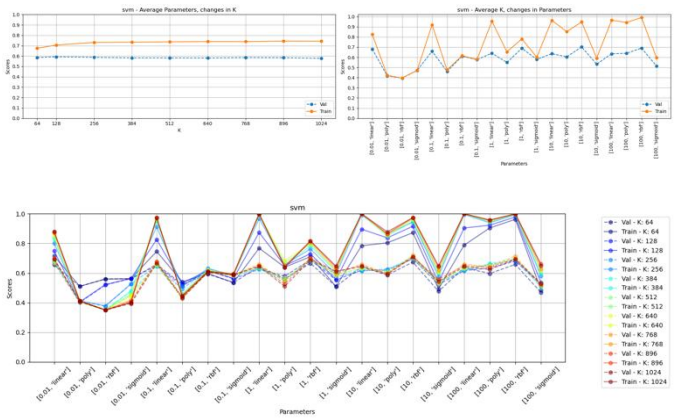
Machine learning model achieves 72% overall accuracy use HOG, specifically, for mono, poly and combined dataset, 79%, 70% and 72% accuracy were obtained. Hog classifier cannot accurately predict class 1 and 2 (33% defect and 66% defect), most of accuracy are gained from class 0 (0% defect) and class 4 (100% defect). Weighted f1 score for best HOG model are 0.77, 0.67, and 0.69, since weighted f1 will consider imbalanced class and assign higher weight to majority class, if average f1 score is calculated for each class, then the result is terrible (This image also stored in code project/out/img).



SIFT also achieved similar result with 72%, 67%^ and 71% accuracy for mono, poly and combined data. 68%, 65% and 67% weighed f1 score are obtained use grid search.



Re-construction method are the worst, it received same accuracy with baseline, 57% accuracy.



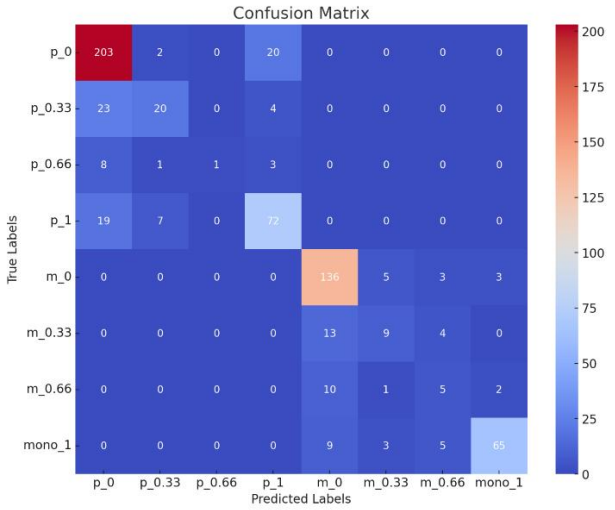
Above plots show SVM classifier under different parameter settings. K value is also considered as a parameter. From first figure, we can see that increasing in k value will slightly increase model performance, and rbf kernel can overfit the model yet also results best validation result.

### *B. Deep Learning*

#### *1) ResNet*

Our model performs well on the test set with high accuracy (up to 80% on average). The results of the confusion matrix show that the model is particularly effective in distinguishing between undamaged and fully damaged solar panels. However, the model's performance in recognizing moderately damaged solar panels (e.g., with damage rates of 33% and 67%) could be improved. This suggests that the model may need further optimization when dealing with finer damage features.

We also use graphs to demonstrate the performance of the model, including accuracy and loss function variation curves. These visualization tools help us to better understand the performance of the model.




| | 8-class | poly | mono | All-type |
|---|---|---|---|---|
| Accuracy | 80.3% | 78.7% | 77.2% | 77.8% |
| Precision | 79.0% | 63.3% | 79.9% | 63.8% |
| Recall | 83.1% | 58.5% | 53.4% | 56.5% |
| F1 Score | 81.0% | 60.3% | 56.0% | 59.1% |

#### *2) MobileNetV3*

```
The total accuracy is: 0.7911585365853658

The accuracy of mono type is: 0.7708333333333334

The accuracy of poly type is: 0.8028846153846154
```

```
The total F1 Score is: 0.607687427092892

The F1 Score of mono type is: 0.5875846549402974

The F1 Score of poly type is: 0.6183270504463467
```

```
The total confusion matrix is:
 [[350  10   1  11]
 [ 35  29   5   4]
 [ 14   5   7   5]
 [ 37   7   3 133]]

The confusion matrix of mono type is:
 [[127   1   1   5]
 [ 17   9   2   1]
 [  5   2   3   2]
 [ 15   3   1  46]]

The confusion matrix of poly type is:
 [[223   9   0   6]
 [ 18  20   3   3]
 [  9   3   4   3]
 [ 22   4   2  87]]
```

## V. DISCUSSION

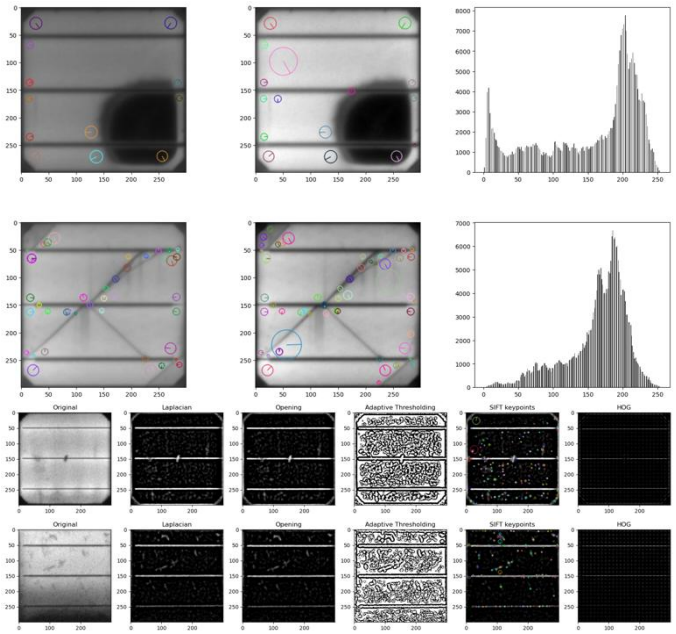
In this section, we will try to analysis the reason why model works or doesn't work.

### *A. Machine Learning part*

The result from machine learning part is unsatisfactory.

#### *1) SIFT Features*

Feature extraction and preprocessing did not enhance classification result or increase model's generalization ability. Use SIFT as example, below two plot shows sift detection result on original image(left), CLACHE enhanced image(middle), from the visualization result, we can find enhancement help us (human) easier capture all the cracks and changes of color, but it SIFT algorithm cannot capture these details (Dark blobs for the first image, crack from second image). From this plot, we can clearly spot few new interests point detected by SIFT, however, with all parameters unchanged, SIFT's performance did not increase at all, SVM performance was reduced for 5%. This could be potentially due to SIFT mechanism. When applying SIFT, we are using these assumptions that defect has its similarities. SIFT is expected to capture similar cracks, blobs, finger cracks, etc. What can SIFT really do? SIFT is robust rich contents, such as: brightness variation, color variation within local window that has certain point of uniqueness that can be captured. An edge is not very interest, since edges is a "thin" change of pixels. This slight change is not unique enough and descriptive enough. Blobs can be a potential interest point, however, to be useful, location and size of blob is crucial. From image analysis, background noises can be variant among different size, shape, and positions. Even if SIFT detected these changes, it might not be easily categorized into same bin, thus make model cannot capture the pattern.




*2) HOG Features*

For HOG descriptor, HOG can capture all the local gradient changes and its directions. Ideally, HOG should capture all changes in the intensity, however, hog cannot capture gradient change diagonally.

Another example of using Laplacian edge detection and adaptive thresholding to find features. Laplacian is very good at detect changes in the image, however, Laplacian is very sensitive to noises. Above images show original image (left one), gaussian blur and Laplacian feature (left 2), morphological opening smooth (left 3), Adaptive thresholding on left 3 (fourth), sift feature on Laplacian image($5^{th}$), and HOG feature on Laplacian image.

*3) SVM*

Another question I have encountered is SVM's class imbalanced issues. Although lots of methods are tested, but the effect is very small. SMOTE only increase accuracy by 1%. When I use ensemble learning bagging, each individual classifier is doing very well, all classifiers can obtain around 99% accuracy, however, when I combine the result together, the accuracy drop dramatically. A wild guess is due to the extreme imbalance.

*4) ICA method*

ICA method is not traditional feature extraction-classification method. It uses the idea sort of like deep learning, where the "error" is calculated. Instead of updating the error, it will just classify result use this error. However, the different calculate uses only the "distance" from basis image and test images. However, a small crack on the image could be in class 1 or 2 even 3, but this can lead to a small error from the calculation, and it will be classifying as non-defect image. This is just a creative method to complete this task.

*5) Paper Comparison*

Compare to original paper, Deitsch, S[1] use two different methods for machine learning's feature extraction. We will focus on our comparison to key points detection. In the paper, Deitsch compared 7 different detector/descriptor, and the best feature detection method is use KAZE as detector and VGG as descriptor. Compare to author's SIFT implementation, author's method peak its f1 score at 75%, and mine was 66%. This match's author's opinion on more grid leads to better results. No HOG metrics was reported. Author also report its best machine learning model KAZE/VGG achieved 88 percent accuracy, and on average 86% use 75% of training samples. That is far better than my method. KAZE and VGG especially separate descriptors and detector methods are worth to dig in more.

## *B. Deep Learning*

*1) ResNet*

In this project, we found that the performance of the ResNet-18 model is limited by the number of samples in the dataset. Although we used oversampling techniques (e.g., SMOTE) to increase the number of samples for some categories, the overall sample size was still low, especially for solar panels with a specific level of damage (e.g., slight or moderate damage). This leads to model limitations in learning and generalizing these specific categories.

Since deep learning models, especially complex ones like ResNet-18, rely on large amounts of data to learn rich and complex features, insufficient data size may result in models that are not adequately trained, which in turn affects their performance in real-world applications. In our case, this may be the main reason for the model's low recognition accuracy on certain damage categories.

In addition, due to the limitation of the amount of data, the model may be overfitted to the training data during the training process and perform poorly on unseen new data. This overfitting is more common in deep learning programs, especially when the available data is limited. However, the use of Convolutional Neural Networks (CNNs) such as ResNet-18 still show good performance. The advantage of CNNs is that they are able to automatically learn complex features from the data, which usually requires manual design and tuning in traditional approaches.

Despite the challenge of insufficient sample size, the ResNet-18 model was able to recognize different levels of solar panel damage. This demonstrates that even under data-limited conditions, the deep learning capabilities of CNNs can provide more accurate classification performance than traditional methods' show their unique strengths in capturing and processing subtle features in images.

*2) Mobile Net*

We introduce mobile NET in this project since it is faster than traditional CNN architecture. Because mobile NET has a lighter weight compared with convolutional neural network (CNN), which indicates faster computation time, but also achieves lower classification accuracy. Our progress is to tune our model so that we can make the most of it, in order to find a way to solve our classification problem.

V1 Setting:

- Output activation function: ReLU
- Optimizer: SGD
- Learning rate: 9e-3
- Loss function: Cross entropy
- Dataset Split: Training with 0.75 and test sets with 0.25 fraction of the dataset.
- Epoch: 20
- Class Distribution: The dataset was imbalanced with class counts of 1136, 222, 75, and 535.

The result of this model has a total 0.77 accuracy and total 0.54 F1 score, which has a higher accuracy but lower F1 score than previous introduce model. As a result, we change the optimizer to Adam, which might improve the F1 score.

V2 Setting:

- Optimizer: Adam
- Learning rate: 1e-5

This model has a total 0.74 accuracy and total 0.44 F1 score, which is still not doing well compared with other models. We try balancing the amount of samples, using SMOTE to enrich the minority classes which is 0.33 and 0.67 damage in our case.

V3 Setting:

- Optimizer: SGD
- Learning rate: 5e-3
- Class Distribution: balanced with 1136 instances in each class.

V3 Setting:

- Optimizer: Adam
- Learning rate: 1e-5
- Class Distribution: balanced with 1136 instances in each class.

The total accuracy two different optimizer are 0.73 and 0.79, respectively. And the total F1 score are 0.56 and 0.60, respectively. Using SMOTE has some positive effect on the model result, but it does not significantly increase the F1 score. We then try using softmax activation function for our output layer.

V4 Setting:

- Optimizer: SGD
- Learning rate: 9e-3
- Epoch: 30

Since we observed that SGD has slightly decrease in accuracy, we then increase epoch to 30, testing whether SGD needs more epoch to get better result. We also increase the learning rate, try to find out a way that can improve the result.

The total accuracy of this model is 0.74, which does not improve much. Moreover, we can see from the above graph that softmax activation model cannot classify the 0.33 and 0.67 damage, leading to a much lower F1 score 0.39.

V4 Setting:

- Optimizer: Adam
- Learning rate: 1e-4
- Epoch: 20

The total accuracy of this model is 0.74 and the total F1 score is 0.44. We can also see from the graph that using softmax activation function at the output layer cannot effectively classify damage between 0.33 and 0.67. Although the correct classified sample in 0.33 damage is not all wrong, this model still cannot consider as an improve as we change the activation function from ReLU to softmax.

We try augmenting the images by using randomHorizontatlFlip, randRotation within 10 degrees and gaussianBlur with kernel_size=3 and possibility=0.5 to test whether the result would be better than the above.

Bad example Setting:

- Data Augmentation: randomHorizontalFlip, randomRotation, and gaussianBlur.
- Optimizer: SGD
- Learning rate: 1e-3
- Epoch: 20
- Loss function: Cross entropy
- Class Distribution: The dataset was imbalanced with class counts of 1136, 222, 75, and 535.

We thought that applying those “augmentations” would train the model to be more adaptive on classifying classes, however, the result does not perform well. The total accuracy of this model is 0.53 and the total F1 score is 0.21, which we consider it as a bad example.

There are some possible reasons why this model has such terrible performance. One is that the data is imbalanced. The amount of 0.33 and 0.67 class samples are 295 and 106, respectively. They represent 0.11 and 0.04 fraction of the whole dataset, respectively. The number of samples in both classes is far from a balanced dataset which is 0.25 in each class,

which could cause the model lean to the minority classes, making it perform poorly on 0.33 and 0.67 classes.

Furthermore, the image might have already been difficult to classify between damaged and undamaged classes. After the "augmentation" process, it only increases the difficulty instead of reducing it. This might also associate with the lack of data. Or maybe we should copy the original data, and then do the augmentation process on the copies. By doing so, increasing the dataset and augmenting the image might improve the training process so that we can get a higher accuracy and f1 score.

## VI. CONCLUSION

In this paper, we start predicting the test set with an ICA model, but it can only detect 0.0 damage and 1.0 damage. We then face the class-imbalanced problem, handling it with SMOTE method, which could generate synthetic samples for minority classes. SMOTE has a positive effect in our case, so we apply this solution in every following approachs. When using SVM classifier, we compared two feature extractors, which are HOG and SIFT. HOG is efficient in object detection and SIFT is extract any descriptor. They both has a great result in f1 score, but their accuracy is not as good as the following models. And then, we introduce ResNet and MoileNetV3, they are based on CNN architecture. The ResNet can avoid "degradation" when the network becomes deeper, and the Mobile Net has lighter weight indicating faster computation time. They both have a high accuracy, but the f1 score is not quite well compared with SVM using HOG and SIFT.

Future improvement

Since all models shows that using SMOTE method can have a positive result in test set, we can further balance the dataset by using ADASYN.

ADASYN[14] is another method we can use to address classes imbalance in classification task. It is suitable for dataset that has a class which is significantly less than other classes. It would generate synthetic samples based on the difficulty in classification, which different from SMOTE that generate samples uniformly.

We can also use ensemble method[15] to improve the result. We can monitor every model in their training process and choose the model which has relatively high accuracy, then based on the high accuracy models' prediction, classifying the result of the specific sample. This would outperform the result when we only use single model for predicting result.

Furthermore, we should mostly increase the amount of dataset. Even we use SMOTE to increase the minority classes in our dataset, we still need more dataset to improve the performance. No matter how we augment or balance the data, we still need adequate data to have a higher accuracy, otherwise models will easily turn overfitting.

## References


[1] Deitsch, S., Christlein, V., Berger, S., Buerhop-Lutz, C., Maier, A., Gallwitz, F., & Riess, C. (2019). Automatic classification of Defective Photovoltaic module cells in electroluminescence images. Solar Energy, 185, 455–468. https://doi.org/10.1016/j.solener.2019.02.067 J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Tsai, D., Wu, S., & Chiu, W. (2013). Defect detection in solar modules using ICA basis images. IEEE Transactions on Industrial Informatics, 9(1), 122–131. https://doi.org/10.1109/tii.2012.2209663

[3] He, H., & Ma, Y. (2013). Imbalanced learning. In Wiley eBooks. https://doi.org/10.1002/9781118646106

[4] Akbani, R., Kwek, S., & Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. In Lecture Notes in Computer Science (pp. 39–50). https://doi.org/10.1007/978-3-540-30115-8_7

[5] Tareen, S. a. K., & Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK. 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET). https://doi.org/10.1109/icomet.2018.8346440

[6] Csurka, G. (2004). Visual categorization with bags of keypoints. *European Conference on Computer Vision*, *1*, 22. https://www.cse.unr.edu/~bebis/CS773C/ObjectRecognition/Papers/Dance04.pdf

[7] Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2016). Feature pyramid networks for object detection. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1612.03144

[8] S. Lazebnik, C. Schmid and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), New York, NY, USA, 2006, pp. 2169-2178, doi: 10.1109/CVPR.2006.68.

[9] Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. and Le, Q.V., 2019. Searching for mobilenetv3. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 1314-1324).

[10] Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[11] Lin, M., Chen, Q. and Yan, S., 2013. Network in network. arXiv preprint arXiv:1312.4400

[12] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for MobileNetV3. arXiv preprint arXiv:1905.02244. Retrieved from https://ar5iv.org/abs/1905.02244

[13] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. arXiv preprint arXiv:1106.1813. Retrieved from https://arxiv.org/abs/1106.1813