Classification of Defective Photovoltaic Cells in Electroluminescence Imagery

MINHUA TANG, KAI CHI MOK, FEI HU, JIALE WANG, XIEYANG JIANG

The University of New South Wales

1. Introduction

As the global shift towards sustainable energy progresses, the utilisation of solar power becomes increasingly vital. Solar panels use photovoltaic (PV) cells to convert sunlight into electrical energy, providing a cost-effective and renewable source of power. However, when exposed to outdoor environments, these panels are prone to damage. Such damage can lead to defects that reduce the panels' efficiency at generating electricity. It is crucial to continuously monitor and assess the integrity of solar panels to optimise their efficiency of energy conversion.

Our project is dedicated to the development of an automated classification system, adept at determining the condition of solar cells through the analysis of electroluminescence (EL) imagery. This system stands to revolutionise the current inspection paradigm by significantly expediting the evaluation process, enhancing accuracy, and superseding labour-intensive and expert-dependent manual inspections.

Central to our investigative endeavour is the ELPV dataset [1]: an extensively curated assemblage of 2,624 EL images, epitomising the benchmark for the visual identification of defective solar cells. Each 300x300 pixel grayscale image is meticulously sourced from a diverse array of 44 solar modules. These images have been subjected to a stringent standardisation protocol, ensuring consistency in size, perspective, and the rectification of optical distortions. Each image is annotated with a probability metric quantifying defect presence: fully functional (0%), possibly defective (33%), likely defective (67%), and certainly defective (100%), and is further categorised based on the type of solar module: monocrystalline or polycrystalline (See Appendix A). In addition to serving as an analytical foundation, this richly annotated dataset also provides us with the opportunity to discern subtle variations in defect probabilities between monocrystalline and polycrystalline cells and to understand how they manifest differently.

This project aims to create robust classification models from training with the ELPV dataset, resulting in a system that is capable of accurately assessing the health (i.e., probability of defectiveness) of solar cells. This initiative will enable meticulous detection and timely intervention when solar panels become impaired, allowing solar energy systems to be operated at guaranteed efficiency with increased longevity of PV modules.

2. Literature Review

The application of computer vision techniques in EL imaging analysis represents a significant advancement in the professional field of electronic device inspection and quality control. Computer vision, a branch of artificial intelligence that trains computers to interpret and understand the visual world, can greatly enhance such analysing capabilities.

The integration of computer vision with solar cell defect detection is a rapidly growing area of research and development. It holds the promise of significantly improving the inspection accuracy and reliability of various solar devices, leading to advancements in solar technology and inspection processes.

2.1 Edge Detection Algorithms

Edge detection algorithms such as Prewitt and Canny are used to detect micro-cracks and edge discontinuities in EL images [2]. Their accuracies are generally high, especially in cases where the image contrast is significant. They are also capable of producing prompt results, which makes them suitable for real-time analysis.

2.2 Image Segmentation Algorithms

Binary thresholding and semantic segmentation are popular choices used to distinguish between defective and non-defective areas in EL images [3,4]. Although their detection accuracy highly depends on the image quality and algorithm parameter settings, their efficiency in computation made them suitable for preliminary screening.

2.3 Pattern Recognition with Machine Learning Algorithms

Support vector machines and convolutional neural networks are common methods utilised to classify defect types as well as to recognise complex defect patterns [5]. These methods are known to have promising accuracies especially when a substantial amount of training data is provided. Despite having a time-consuming training process, the detection speed is almost instant once trained.

2.4 Challenges in Photovoltaic Cell Inspection

Dependence on Image Quality: The quality of EL images is related to the performance of computer vision algorithms [6]. Noise, contrast, resolution, and lighting conditions are among the important contributing factors. Addressing this challenge requires high-quality imaging equipment and advanced image preprocessing techniques.

Algorithm Complexity: Advanced machine learning and deep learning algorithms, while highly accurate, are complex and demand significant computational resources [5]. This can lead to increased processing times, especially for a large dataset.

Generalisability: Computer vision models need to work effectively across various EL images, but their generalisability is limited by the diversity and quantity of training data [3,5]. Collecting and annotating a large, high-quality dataset is challenging [6].

Real-time Monitoring and Automation: In real-world production environments, real-time monitoring and automated detection are key requirements, necessitating algorithms to be not only accurate but also fast, efficient, and portable [5].

In summary, the application of computer vision technology in EL imaging analysis holds enormous potential, but achieving optimal performance requires overcoming challenges related to image quality, algorithm complexity, and continuous optimisation of algorithms to suit various environments.

In this project, we will focus on exploring machine learning and deep learning methods together with various image data processing techniques to achieve effective classification of PV cell defects.

3. Methodology

Upon scrutinising the dataset, our intuition suggests that training without pre-processing the data first might simply be insufficient for building a good-performing classification pipeline. For that reason,

a curated set of pre-processing techniques and classification models are explored in the hope of improving prediction accuracies. The subsequent discussion provides an overview of the selected methods for implementation (See Appendix B) and the reasons guiding their choices.

3.1 Classing

An initial challenge arises regarding determining the classes and the optimal quantity thereof. The dataset contains 3 subclasses of information: images, defect probabilities, and types of solar wafers.

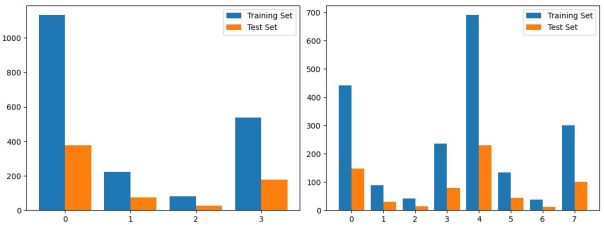
4-Class is a relatively straightforward classing approach. It involves categorising images based on a single metric, the defect probability, alone. The 4 discrete defect categories (i.e., 0%, 33%, 67%, 100%) are mapped to 4 distinct numerical class labels (i.e., 0, 1, 2, 3). Despite the ease of mapping, this approach is likely to produce an unequal partition of mono- and poly-crystalline data across the training and validation subsets after splitting. It may also exert an impact on distinguishing mono- or poly-specific defects.

8-Class expands upon the 4-class model by including the type of solar module as an additional parameter for classing. This means each image is classified not only by its defect probability but also by its module type. Monocrystalline samples are mapped to the first 4 classes (i.e., 0, 1, 2, 3) in accordance with the 4-class approach while the remaining classes (i.e., 4, 5, 6, 7) are allocated for polycrystalline samples ordered by defect category. This additional differentiation may capture more photovoltaic cells' characteristics and may exert a better accuracy at classification tasks, which remain to be verified through experiments.

3.2 Dataset Partitioning

Once the classes are defined, it becomes pivotal to ascertain an appropriate cutoff ratio for partitioning the dataset into two subsets for training and testing purposes.

Empirical research by A. Gholamy et al. [7] indicates that optimal outcomes are achieved when 20 – 30% of the data are allocated for testing and the remaining 70 – 80% are reserved for training. Consequently, the mid-points of these cutoff values (i.e., 75 / 25) are chosen as our final split ratio.



Stratified-Shuffled-Split: Examples of partitioning the 4-class or 8-class ELPV dataset into 2 subsets.

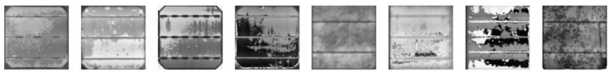
With the 75 / 25 split, the training set contains 1,968 samples which occupied 75% of the total samples, and the test set contains 655 samples which occupied 25% of the total samples. During the splitting process, stratification and random shuffling are employed to ensure data are randomly distributed among the 2 subsets with a roughly equal proportion of different classes.

3.3 Oversampling

Class imbalance becomes a significant issue once EL images are mapped to their respective classes. It is suspected that an unequal distribution of samples across various classes might exert an influence on the training outcomes. To address this issue, we strategically employ various over-sampling techniques to artificially augment the training data size of minority classes.

Each of the following techniques synthesizes new minority class samples from its own class but with nuanced approaches tailored to various aspects of the dataset to enhance performance.

SMOTE: Generates new synthetic samples for minority classes instead of naïve duplication to mitigate bias [8]. In the context of EL image classification, this method is particularly useful for capturing crucial information about minority cells.



SMOTE Synthetic Samples: Examples of synthetic samples generated by the SMOTE Over Sampler.

SVMSMOTE: Uses a Support Vector Machine to identify samples near the decision boundary (i.e., the most difficult to classify). New synthetic samples are then generated along the boundary to create a more distinguishable separation between classes. This method is particularly effective when dealing with complex and overlapping class distributions, as it focuses on the regions where the classifier is most likely to make mistakes.

BorderlineSMOTE: Specifically focuses on samples that are on the borderline between two classes [9]. It generates synthetic samples near these borderline instances. The rationale is that improving the classification of borderline instances can significantly enhance the overall classifier performance, given that these instances are often the most challenging ones to classify correctly. By focusing on these critical regions, BorderlineSMOTE can lead to a better-defined decision boundary in the model.

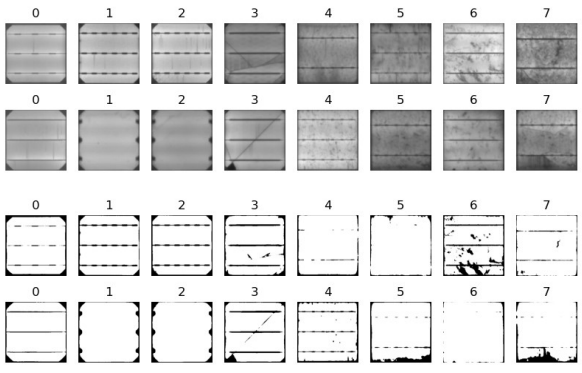
The above methods were selected based on their proven efficacy in addressing dataset imbalance [8,9]. By enriching our training data with synthesis techniques, we aimed to significantly improve our model's ability to accurately classify and predict the health of PV cells in EL images. Note that by convention, the test set or validation set is not resampled as it might introduce bias in evaluation [10].

3.4 Class-Specific Thresholding

Straight-forward methods for extracting defect features from EL images are to be employed in our implementation to evaluate their impact on classification performance [11].

BINARY + OTSU for Monocrystalline Cells:
This thresholding technique is used for mono cells (Class 0 – 3) to separate the foreground (i.e., defective areas) from the background more effectively, enhancing the feature distinction in EL images.

TOZERO ⟶ BINARY + OTSU for Polycrystalline Cells:
Polycrystalline cells (Class 4 – 7) have a different texture and structural characteristics compared to mono cells. Therefore, a two-step thresholding process is employed. The TOZERO method initially isolates specific intensity ranges, and The BINARY + OTSU method that follows clears the demarcation of defects.



Before and After Thresholding: The first 4 classes contain Monocrystalline samples with defect probability of 0, 0.3, 0.6 and 1 respectively. The remaining 4 classes contain Polycrystalline samples in the same order of defect probability.

A class-specific approach ensures that the largest extent of defect features is extracted from EL images concerning the conditions and traits they possess. Edge detection is another alternative.

3.5 Classification Using General-Purpose Classifiers

In this project, we employ various general-purpose classifiers to predict the health of PV cells in EL images. This approach aims to evaluate and compare the performance of different classifiers in identifying varying degrees of functional and defective cells.

Stochastic Gradient Descent (SGD): Effective for large-scale and sparse data, providing robustness in handling noise and outliers.

Decision Tree (DT): Offers clear visualisation of the decision-making process, aiding in understanding the feature's importance in classification.

K-Nearest Neighbours (KNN): Utilises proximity to neighbouring data points for classification, beneficial in capturing local patterns within the data.

Gaussian Naïve Bayes (GNB): Assumes independence among predictors, suitable for high-dimensional datasets [12].

Multinomial (MNB) and Bernoulli Naïve Bayes (BNB): These are variations of Naïve Bayes, tailored for multinomially and binary distributed data, respectively.

Multi-Layer Perceptron (MLP): A type of neural network classifier capable of capturing complex relationships in the data through its layered structure.

Support Vector Machine (SVM): Known for its effectiveness in high-dimensional spaces and its versatility through various kernel functions [13].

The choice of these classifiers and the specific techniques for feature extraction and pre-processing are driven by the need to achieve high accuracy and efficiency in classifying solar cell defects. This ensures the health of PV cells from EL imagery is monitored accurately, contributing to the optimal functioning of solar panels.

3.6 Classification Using Deep Convolutional Networks

Apart from utilising traditional classifiers, 2 deep convolutional neural networks specialised in image classification are employed to train with the ELPV dataset. The approach aims at evaluating and comparing their difference in performance under a different network structure and when used in conjunction with different data pre-processing techniques mentioned above.

3.6.1 Data Augmentation as ConvNet Block

In the pursuit of enhancing the accuracy of our model, we recognised that the dataset of 2,624 images was insufficient for robust training and testing.

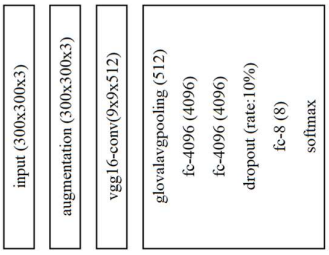
We employ data augmentation as a technique to generate additional, slightly modified training samples. Since the EL images from the dataset are already normalised with respect to size and perspective and corrected for camera lens distortions, we would only introduce a moderate level of augmentation so that the augmented samples are in line with such conditions. Augmentation is implemented as a module block in the ConvNet comprising several processing layers. We adopt the following well-tested data augmentation scheme by S. Deitsch et al. [5]: Vertical and horizontal flips are done at random. Random rotation is limited to $\pm 3.6^{\circ}$. Random zooming is restricted to $\pm 2\%$ with aspect ratio preserved. Random translation is employed to shift EL images at a height and width factor of $\pm 2\%$.

3.6.2 Modified VGG16 with ImageNet Transfer Learning

Apart from augmenting data, transfer learning is also something in our first thought when insufficient data comes into play. The motivation here is to largely exploit the well-established VGG16 [14] architecture pre-trained with the data-abundant general-purpose ImageNet dataset. Having weights in our deep neural network pre-trained with over 1.28 million colour images and 1000 classes, the model is believed to perform as promising as it is when applied to the domain-specific ELPV dataset as well.

The approach here is to avail the use of VGG16 frameworks and the associated pre-trained weights from ImageNet but replace the fully connected layers with our own implementation. With inspiration from the modified VGG19 model proposed by S. Deitsch et al. [5], we implement a similar modification to the block by incorporating a special layer of Global Average Pooling (GAP) followed by 3 fully connected layers of 4,096, 4,096 and 8 neurons, respectively. Given that we have a size of 300x300x3 EL input tensor instead of VGG16's default 224x224x3 tensor [14], the GAP layer reduces the need for further down-sampling and enhances the compatibility of our EL image dataset to VGG16's default configuration [5].

Modified VGG16 Configuration: The convolutional layers are denoted as "conv<kernel size>-<number of filters> (output shape)".

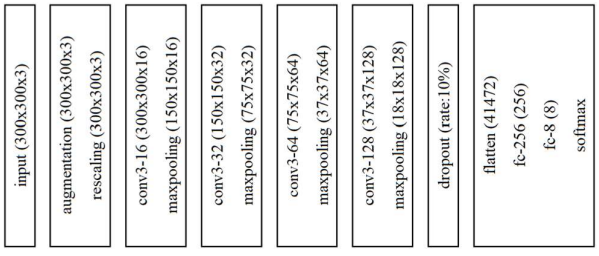


Since VGG16 is regarded as one of the best-performing ConvNet models for image classification, it is chosen as our base reference model for evaluating the performance of our handcrafted variant of VGG (to be discussed in the following section). The pre-trained weights in VGG16's convolutional layers are not to be trained during training so that we can fully exploit the computation gain on transferred knowledge. The augmentation and dropout layers are added to reduce overfitting.

3.6.3 Simplified VGG (HC6) with Randomly Initialised Weights

VGG16 comprises several stacks of convolutional layers each with a considerable number of filters. Together with striding in the max-pooling layers, the entire framework requires substantially more time (e.g., several hours) to compute and train the model. Our time constraint disallows us to recursively experiment with such a time-consuming model alongside other data pre-processing techniques in a time-effective manner. Despite VGG16 being a well-proven model, its high performance is at the expense of heavy computational costs. We are in need of an experimental model capable of completing the training within a shorter time span (e.g., quarters of an hour).

Concerning this, we are motivated to handcraft our own variant of VGG, known as Handcrafted-6, from scratch with fully trainable and randomly initialised weights. The goal is to evaluate whether a simplified framework can achieve a performance as close as VGG16 under a limited context. If that is proven, we can safely employ this model to effectively simulate the behaviour of VGG when used in conjunction with multiple data preprocessing techniques.



HC6 (Handcrafted-6) Configuration: The convolutional layers are denoted as "conv<kernel size>-<number of filters> (output shape)". The ReLU activation functions are not shown for brevity.

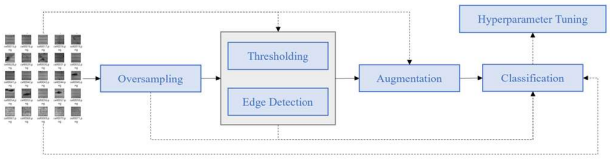
The 6-weight-layer model is derived from a collection of VGG frameworks outlined in the ICLR paper by K. Simonyan et al. [14]. It captures the idea of having multiple stacks of channel-increasing but tensor-decreasing convolutional blocks while the network depth increments. The model, however, simplifies such configuration with fewer layers and filter channels on each of the blocks. The model also eliminates the need to perform striding in each of the pooling

layers. A significant reduction in computation overhead not only creates room for repeated experiments but also enables the ease of adjustment and observation of the model performance.

Instead of a continuous value, the final output is a normalised class score map with a vector size of 8, each of which corresponds to the confidence level for that class prediction.

4. Experimental Setup And Results

Our experiments are divided into 2 major pipelines, within which a different classification model is used in conjunction with none or some pre-processing techniques for investigating performance gain.



Experiment Pipelines: An overview of the classification pipeline and their associated variations.

Our experiments are conducted on Anaconda, a popular platform for conducting machine learning and data science studies. Methods and codes are written in Python 3 standards with the use of the following applications and libraries:

Environment	Anaconda 3 Python 3.10.13
Application	Jupyter Notebook 6.5.4
Libraries	TensorFlow 2.10.0, Keras 2.10.0, scikit-learn 1.2.2, imbalanced-learn 0.10.1, OpenCV 4.6.0, pillow 10.0.1, NumPy 1.26.0, matplotlib 3.8.0

Note that both setups used the same split of the dataset on all its iterations to ensure fairness in evaluation. Test set predictions are evaluated based on the accuracies, precisions, recalls, F1-scores and 4x4 confusion matrices computed first with the 2 solar module types combined and then separated (mono followed by poly). Metrics are weighted while matrices are ordered for ease of evaluation.

The following presents the setups used for assessing the performance of our developed implementations and their respective results.

4.1 General-Purpose Classifier Algorithms

In this approach, we trained 8 machine learning algorithms with the ELPV dataset, examined the performance, used in conjunction with several pre-processing techniques to assess any performance gain, and selected the best-performing pipeline to conduct a comparative study with ConvNet's performance. In some of the pipelines, pre-processing methods, such as oversampling, thresholding, and edge detection, are applied to the given dataset before training. The goal is to compare performance across different classifiers and explore addon combinations that enhance accuracies. To summarise, there are in total 6 iterations of model training illustrated as follows:

	Thresholding	Laplacian Enhancement	Oversampling	Hyperparameter Tuning	Model Training
4.1.1					✓
4.1.2	✓				✓
4.1.3		✓			✓
4.1.4			✓		✓
4.1.5	✓		✓		✓
4.1.6	✓		✓	✓	✓

The ELPV dataset obtained from GitHub [1] is initially loaded into the memory using a slightly modified version of "elpv_reader.py", a dataset loader provided by the asset author [1]. Implemented with Pillow, the modified dataset loader retrieves the 2,624-sample dataset from Jupyter Notebook's root directory instead of elsewhere.

Upon completion of data retrieval, we utilised a 4-class approach in categorising the data with respect to the provided annotations. The 4 class labels are 0.0, 0.33, 0.67 and 1.0, each corresponding to 1 of the 4 defect probability categories.

```
proba = proba.astype(str)
x_train, x_test, y_train, y_test = train_test_split(images, proba, test_size=0.25, stratify=proba)
```

Dataset Splitting: An overview of the implementation with scikit-learn train_test_split method.

Scikit-learn train_test_split method is being implemented to randomly split the image data into the training and testing subsets. The training set contains 1,968 samples, which accounted for 75% of the total samples, and the testing set contains 655 samples, which occupied 25% of the total samples. The parameter 'stratify' is being used on defect probability, which enabled the same distribution of the 4-class data across the training and test set.

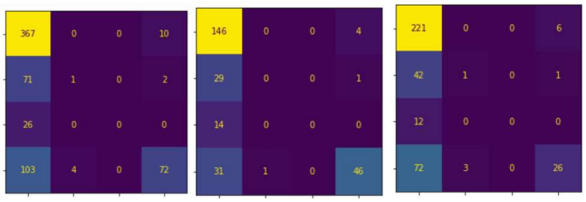
The test set is further split into 2 subsets with respect to solar wafer types: monocrystalline and polycrystalline, which are being used to assess the model performance compared with the full test set.

4.1.1 Model Training without Pre-processing

There are 8 supervised machine learning algorithms employed to examine the performance of traditional classifiers, including Linear Stochastic Gradient Descent, Decision Tree, K- Nearest Neighbours, Gaussian Naïve Bayes, Multinomial Naïve Bayes, Bernoulli Naïve Bayes, Multi-Layer Perceptron and Support Vector Machines.

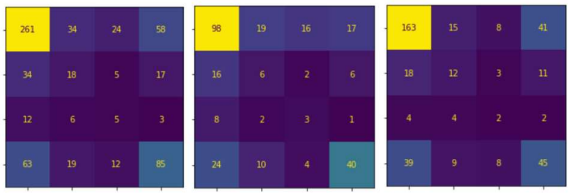
In this iteration, no pre-processing or hyperparameter tuning is implemented. The main goal is to find out the optimal base model for the classification task so that we can use it as a reference for further evaluation in other iterations.

4.1.1.1 Stochastic Gradient Descent



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.67	0.71	0.65
Precision	0.63	0.62	0.61
Recall	0.67	0.71	0.65
F1-Score	0.60	0.64	0.56

4.1.1.2 Decision Tree

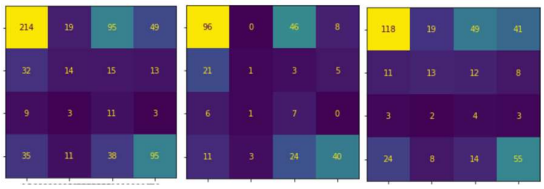


	Combined	Monocrystalline	Polycrystalline
Accuracy	0.56	0.54	0.58
Precision	0.58	0.57	0.59
Recall	0.56	0.54	0.58
F1-Score	0.57	0.55	0.58

4.1.1.3 K- Nearest Neighbours

	Combined	Monocrystalline	Polycrystalline
Accuracy	0.66	0.65	0.66
Precision	0.63	0.62	0.63
Recall	0.66	0.65	0.66
F1-Score	0.63	0.63	0.68

4.1.1.4 Gaussian Naive Bayes

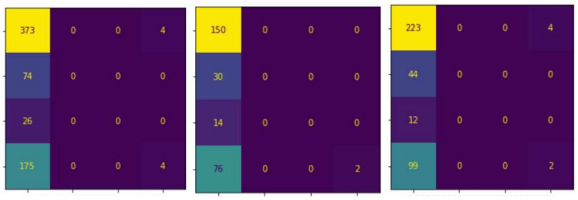


	Combined	Monocrystalline	Polycrystalline
Accuracy	0.51	0.53	0.49
Precision	0.62	0.64	0.62
Recall	0.51	0.53	0.49
F1-Score	0.55	0.56	0.54

4.1.1.5 Multinomial Naive Bayes

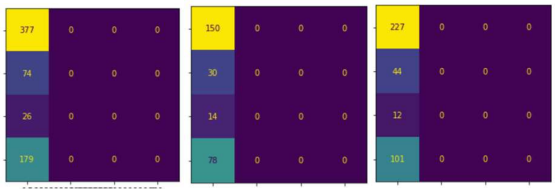
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.38	0.29	0.45
Precision	0.61	0.60	0.60
Recall	0.38	0.29	0.45
F1-Score	0.45	0.34	0.49

4.1.1.6 Bernoulli Naive Bayes



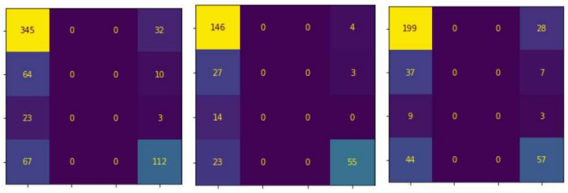
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.57	0.56	0.59
Precision	0.47	0.59	0.44
Recall	0.56	0.56	0.59
F1-Score	0.43	0.41	0.45

4.1.1.7 Multi-Layer Perceptron



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.57	0.55	0.59
Precision	0.33	0.30	0.35
Recall	0.57	0.55	0.59
F1-Score	0.42	0.39	0.44

4.1.1.8 Support Vector Machines



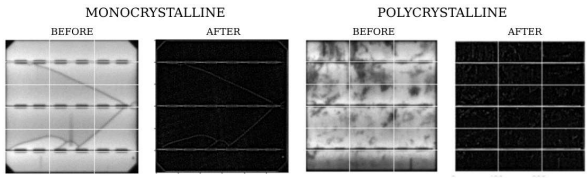
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.70	0.74	0.67
Precision	0.59	0.64	0.56
Recall	0.70	0.74	0.67
F1-Score	0.63	0.67	0.61

4.1.2 Laplacian Image Sharpening

Given that the samples are grayscale images, if we can enhance the visible texture in the image with image sharpening techniques, we might be able to detect the defect edges within the given image and hence improve the performance [15] in classifying defective cells. Here, we applied Laplacian sharpening with the following kernel:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 8 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

As we have 2 types of solar panels, the images before and after Laplacian enhancement are presented as follows:



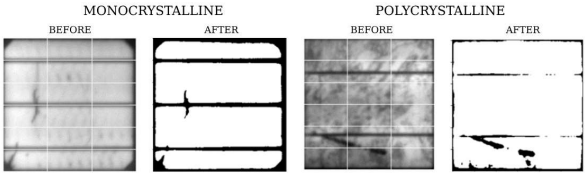
Based on Section 4.1.1, SVM is the best-performing model among the 8 experimented classifier models. Therefore, from this iteration onwards, we will focus on experimenting with different preprocessing techniques with SVM and propagating the best-performing pipeline to the next iteration. The result of this iteration is as follows:

	Combined	Monocrystalline	Polycrystalline
Accuracy	0.66	0.73	0.60
Precision	0.55	0.63	0.50
Recall	0.66	0.73	0.60
F1-Score	0.59	0.67	0.54

4.1.3 Thresholding

Given that the defects on the grayscale solar panel images are visible as broken cracks or dark spots, if we can enhance the contrast level with some image thresholding techniques, it might help the model identify the difference among the 4 defect classes more easily.

In this iteration, we utilised 2 different streams of thresholding methods based on the types of solar panels with the expectation of enhancing the performance gain. For monocrystalline cells, we have implemented OTSU thresholding while for polycrystalline panels, we have implemented TOZERO thresholding followed by OTSU thresholding. Examples of images before and after class-specific thresholding applied are presented as follows:



The performance of the SVM pipeline having the above class-specific thresholding methods incorporated are as follows:

	Combined	Monocrystalline	Polycrystalline
Accuracy	0.70	0.73	0.60
Precision	0.71	0.63	0.62
Recall	0.70	0.73	0.60
F1-Score	0.65	0.67	0.46

4.1.4 Oversampling

The given samples are imbalanced as illustrated below:

```
from collections import Counter
print(sorted(Counter(y_train).items()))
print(sorted(Counter(y_test).items()))
```

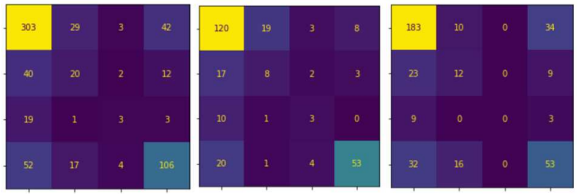
```
[('0.0', 1131), ('0.3333333333333333', 221), ('0.6666666666666666', 80), ('1.0', 536)]
[('0.0', 377), ('0.3333333333333333', 74), ('0.6666666666666666', 26), ('1.0', 179)]
```

In this iteration, we mainly implemented the oversampling methods to the training dataset and examined the performance gain of a balanced dataset on SVM training. There are in total 3 oversampling methods being used in this iteration, namely SMOTE, SVMSMOTE and BorderlineSMOTE from scikit-learn.

4.1.4.1 SMOTE Oversampling

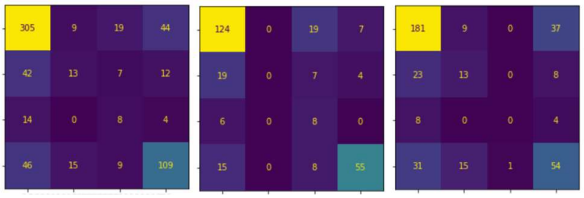
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.67	0.71	0.64
Precision	0.63	0.65	0.60
Recall	0.77	0.71	0.64
F1-Score	0.64	0.67	0.62

4.1.4.2 SVMSMOTE Oversampling



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.66	0.68	0.65
Precision	0.64	0.68	0.61
Recall	0.66	0.68	0.65
F1-Score	0.65	0.67	0.63

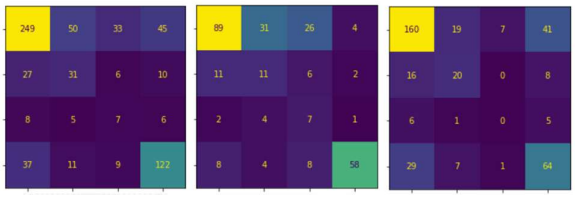
4.1.4.3 BorderlineSMOTE Oversampling



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.66	0.69	0.65
Precision	0.65	0.67	0.62
Recall	0.66	0.69	0.65
F1-Score	0.65	0.67	0.63

4.1.5 Thresholding + Oversampling

Based on experiment iterations from 4.1.1 to 4.1.4, we can see that thresholding slightly improves the performance for the mono-crystalline defects classification, whereas SMOTE, in particular, improves the performance for the poly type of classification without compromising the combined performance much. For that reason, in this iteration, we combine SMOTE with the best-performing pipeline from the previous iteration and evaluate the performance.



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.62	0.61	0.64
Precision	0.67	0.73	0.64
Recall	0.62	0.61	0.64
F1-Score	0.64	0.65	0.64

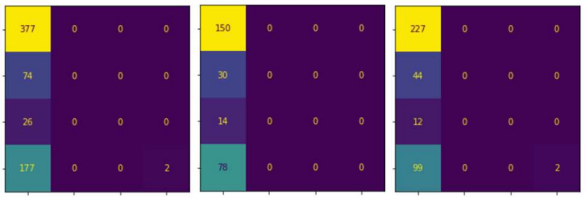
4.1.6 Thresholding + Oversampling + Hyperparameter Tuning

As in all our experiments above, we have not implemented any hyperparameter tuning which is a commonly used way to improve model performance [16]. With respect to this, we have set up a grid search in this final iteration of experiment to find out the optimal combination of parameters for our best-performing model.

However, due to the huge amount of time grid search consumes, we have limited the scope of experiment to only include the following parameters which end up with no overall performance gain:

```
C:[0.1, 1]  gamma:[0.1, 1]  kernel: [rbf]  cv: 2
```

Based on the above configuration to reduce the model running time, the refined model performance is as follows:



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.58	0.55	0.60
Precision	0.60	0.30	0.61
Recall	0.58	0.55	0.60
F1-Score	0.43	0.39	0.45

4.2 Deep Convolutional Networks

In this approach, we train the two deep convolutional networks: VGG16 and HC6, with the ELPV dataset obtained from GitHub [1] to comparatively examine the performance of ConvNets over traditional classifiers in categorising defective solar panels. Apart from investigating the impact Transfer Learning and ConvNet structure would have exerted on performance, we also want to observe the impact on performance if we pre-process the data further prior to any training. To achieve this motive, we make use of our handcrafted model HC6 to repeatedly simulate the result of what a typical VGG would have behaved when employed in conjunction with techniques to preprocess data before training. The following summarises the experiments conducted:

	OVERSAMPLING	AUGMENTATION	THRESHOLDING	MODEL
4.2.1				VGG16
4.2.2.1				HC6
4.2.2.2	✓			HC6
4.2.2.3		✓		HC6
4.2.2.4			✓	HC6

Unlike general-purpose classifiers which are used to experiment with several combinations of techniques, the focus for this pipeline is to experiment with ConvNet construction and observe the behaviour with only one level of pre-processing each time. Given that ConvNet training generally takes more time, we are unable to assess a variety of combinations like Section 4.1 due to time constraints.

The ELPV dataset is initially loaded into our Jupyter Notebook environment using a modified version of "elpv_reader.py", a dataset loader that comes with the dataset [1]. Implemented with Pillow, the modified loader loads the full 2,624-sample dataset and converts image data into 3 channels (RGB) instead of 1 channel (grayscale). The reason for this is due to VGG16's configuration which requires 3 channels (RGB) for each input tensor. Instead of retrieving a 300x300 2D vector for each image data, we obtain a 300x300x3 3D vector which VGG16 is compatible with, and that facilitates the training process in a later stage of the experiment. The modified loader also maps each of the EL images to its corresponding class label using an 8-Class approach.

The loaded data is then shuffled, stratified, and split into 2 subsets - training set and test set, at a proportion of 7.5 : 2.5 (75 / 25 approach: 1,968 training samples, 656 test samples) by utilising a scikit-learn assisted implementation.

If oversampling is a testing factor in that experiment, the training set is resampled to a size of 5,520 samples using oversampling techniques provided by the imbalanced-learn library. The sampler is configured to use a "not majority" policy, which enables minority classes to be expanded to the first majority class level with synthetic samples of their own class. Note that the test set is not resampled.

If thresholding is a testing factor in that experiment, the image data is pre-processed using a set of class-specific thresholding techniques mapped specifically to the module type it belongs. Edge detection is not to be included in this pipeline as external feature extraction is not the focus. Thresholding is implemented using OpenCV methods.

Once the data has been thoroughly pre-processed, the training samples in batches of 32 are used to train the neural network for a maximum of 50 epochs. During the training process, test samples are used as the validation data to monitor the training performance. Implemented with Keras and TensorFlow in the backend, the 2 deep ConvNet frameworks are constructed in accordance with the layout and configuration as specified in Section 3. Note that after model construction, our modified VGG16 has over 18 million trainable parameters while our simplified model HC6 has 10 million. VGG16 requires an additional model-specific preprocessing, which subtracts the mean RGB value from each pixel of the training data [3].

Given that data augmentation is part of the network layer, during the training process it randomly creates 5-fold slightly augmented versions of the original EL samples to refine the model. Note that the image data are conventionally scaled down to a range between 0 – 1 from 0 – 255 before entering the convolution.

Here, we employ the Adam optimiser and Sparse Categorical Cross-entropy loss function to refine the weights of the model. Parameters of the 2 functions are compiled into the model with a configuration well-tested by S. Deitsch et al. [5]. In this respect, the model is trained at a learning rate of 10^{-3}, exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and epsilon value of 10^{-8}.

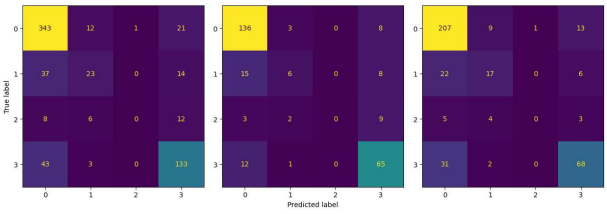
The training procedure is run on NVIDIA GeForce RTX 3060 with early-stopping and checkpoint mechanisms in place. The early-stopping function monitors the validation accuracy and halts the training if the metrics stop improving after 10 epochs. The checkpoint function saves the best-performed model having the best validation accuracy during the training process.

Hyperparameter refinement is not to be observed in this experiment given the time constraint and computational cost if it were to be implemented with ConvNets.

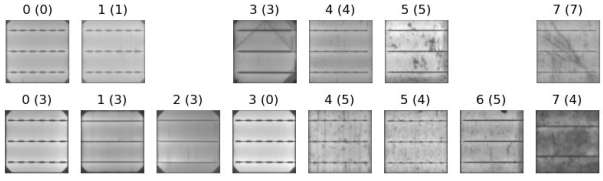
The final prediction of an EL input is computed from a vector of 8 known as the class score map, each of which corresponds to a SoftMax-normalised confidence probability for that class prediction. Our algorithm simply picks the class with the largest magnitude as our final prediction for that sample. Given there are rarely (i.e., within 2%) samples wrongly predicted across module types, we combined the result metrics / matrices from 8 classes to 4 classes for ease of evaluation.

4.2.1 VGG16 with Transfer Learning

In this iteration, we trained the model with pre-trained weights from ImageNet and without any pre-processing. The model has a very balanced performance with a good overall accuracy (76%) and only a small bias (0.02) among the 2 solar module types.



	COMBINED	MONOCRYSTALLINE	POLYCRYSTALLINE
ACCURACY	0.76	0.77	0.75
PRECISION	0.72	0.75	0.72
RECALL	0.76	0.77	0.75
F1-SCORE	0.73	0.73	0.73



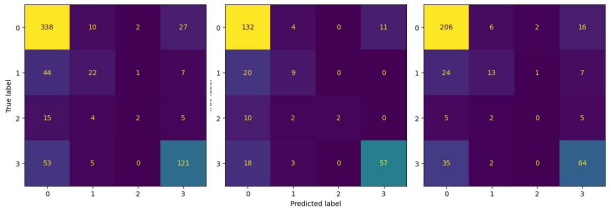
Examples: The first and second row corresponds to the first correctly and incorrectly classified EL samples for each class. The correct class label is denoted by (<label>).

4.2.2 HC6 without Transfer Learning

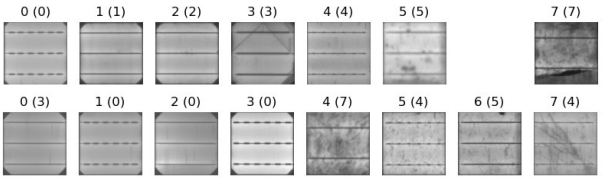
HC6 is employed given VGG6's computational expense.

4.2.2.1 Direct Training

The result shows that our handcrafted model behaves almost as good as VGG16 (0.74 vs 0.76) under a domain-specific context, and hence is sufficient for quickly simulating VGG behaviour in this project.



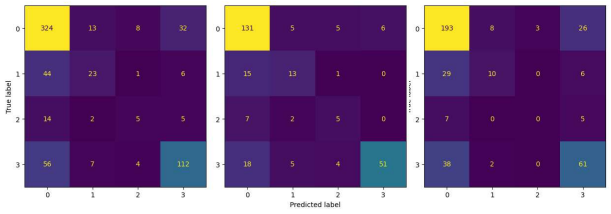
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.74	0.75	0.73
Precision	0.71	0.75	0.70
Recall	0.74	0.75	0.73
F1-Score	0.71	0.72	0.71



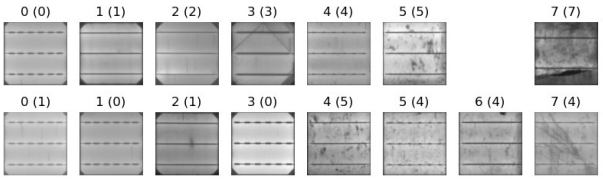
Examples: The first and second row corresponds to the first correctly and incorrectly classified EL samples for each class. The correct class label is denoted by (<label>).

4.2.2.2 Oversampling: SMOTE

The result implies that oversampling does not necessarily improve accuracy as more bias (0.07) is introduced among the 2 wafers.



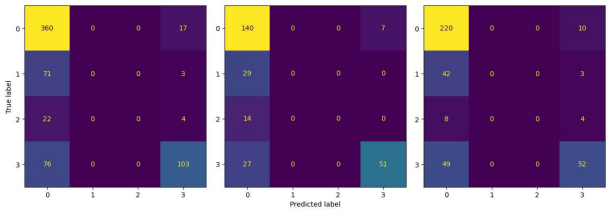
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.71	0.75	0.68
Precision	0.69	0.75	0.65
Recall	0.71	0.75	0.68
F1-Score	0.69	0.74	0.66



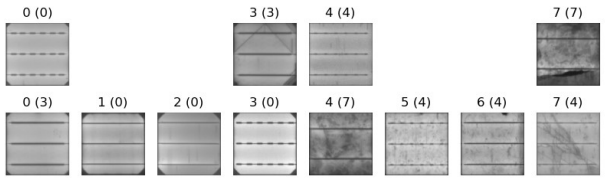
Examples: The first and second row corresponds to the first correctly and incorrectly classified EL samples for each class. The correct class label is denoted by (<label>).

4.2.2.3 Augmentation

The drop in performance implies that augmentation does not necessarily improve accuracy but introduces further degradation.



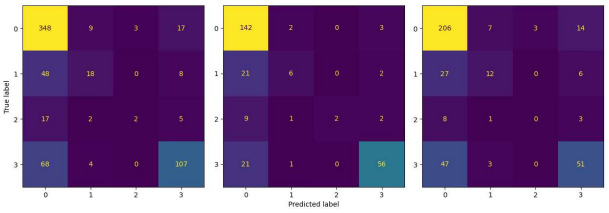
	Combined	Monocrystalline	Polycrystalline
Accuracy	0.71	0.71	0.70
Precision	0.72	0.74	0.71
Recall	0.71	0.71	0.70
F1-Score	0.64	0.65	0.63



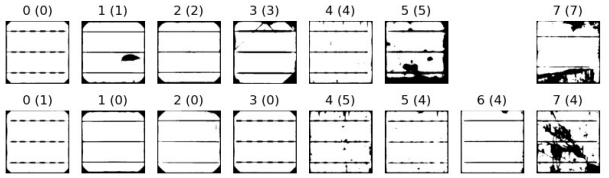
Examples: The first and second row corresponds to the first correctly and incorrectly classified EL samples for each class. The correct class label is denoted by (<label>).

4.2.2.4 Class-Specific Thresholding

Thresholding might enlarge the bias gap (0.08) among the 2 wafer types without making much improvement to performance.



	Combined	Monocrystalline	Polycrystalline
Accuracy	0.72	0.77	0.69
Precision	0.71	0.78	0.66
Recall	0.72	0.77	0.69
F1-Score	0.69	0.74	0.66



Examples: The first and second row corresponds to the first correctly and incorrectly classified EL samples for each class. The correct class label is denoted by (<label>).

5. Discussion

Based on the results presented in Section 4 for the 2 experiment pipelines implemented, we will, in this section, analyse in detail the classification performance and findings from different perspectives.

5.1 Model Performance per Ground Truth Type

The following evaluates the performance of our models by solar module types, defect categories and the classing approach.

5.1.1 Module Type: Monocrystalline vs Polycrystalline

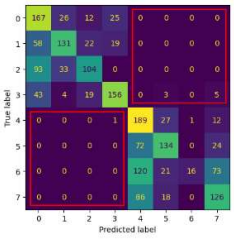
The performance differences in monocrystalline and polycrystalline panels' defect detection largely vary with the methods used. While there is a tendency for defects in mono cells to be classified more accurately than poly cells in ConvNets, some iterations in the classifier pipeline suggest the opposite. There are also occasions where the 2 types of cells do not differ by much.

While our analysis cannot draw a conclusive finding on this aspect, it is important to recognise the physical differences between the 2 types of panels. Monocrystalline panels, known for their higher efficiency and cost due to the high-purity silicon used, might exhibit different defect characteristics compared to the less efficient but more affordable polycrystalline panels [6]. The limitations of our dataset, particularly in terms of diversity and size, might have contributed to such variations in performance. Future studies could benefit from a larger and more varied dataset, possibly encompassing different geographic locations and a broader range of environmental conditions.

5.1.2 Defect Category: 0 vs 0.3 vs 0.6 vs 1

From the presented confusion matrices, our models appear to perform the worst for samples that fall within 0.33 and 0.67 defect categories. There are multiple occasions where our models failed to correctly classify any EL images into these 2 categories, resulting in a class recall rate of as poor as 0. A closer look at the dataset reveals that 0.3 and 0.6 are among the two smallest minority classes in terms of sample size. That explains why our model performance cannot go beyond 80% given that our data-hungry models are insufficiently trained to oversee defects of these minority categories. A solution is to expand the dataset with more samples to the 2 imbalanced classes.

5.1.3 Classing: 4-Class vs 8-Class



The difference in performance for models using a 4-class and 8-class approach in training can be neglected because there are rarely samples misclassified to the incorrect solar wafer types (highlighted in red square). This might be due to the fact the 2 solar panel types have significant differences in appearance which a non-expert human can also easily distinguish.

5.2 Model Performance per Oversampling Method

As discussed in Sections 3 and 4, the given samples from 4 classes are extremely imbalanced. To improve the model performance, various oversampling methods are being employed in our 2 experiment pipelines, including SMOTE, SVMSMOTE and BorderlineSMOTE. A summary of the results is provided as follows:

	SVM				HC6 (CNN)	
(Overall)	Origin	SMOTE	SVMSOMTE	Borderline SMOTE	Origin	SMOTE
Accuracy	0.70	0.67	0.66	0.66	0.74	0.71
Precision	0.59	0.63	0.64	0.65	0.71	0.69
Recall	0.70	0.67	0.66	0.66	0.74	0.71
F1-Score	0.63	0.64	0.65	0.65	0.71	0.69

5.2.1 Oversampling vs No Oversampling

By observing the model performance of the results with and without applying oversampling methods, it is found that there is no positive improvement in overall performance when applying oversampling on ConvNets. However, oversampling methods are found to have improved the precision and F1-score performance for SVM with the compromise of accuracy and recall performance. The best-performing oversampling method in this specific context is SMOTE, but not its other variants.

An explanation is that the raw dataset might have been sampled with the concept of likelihood in mind. In other words, some samples might appear less frequently in real-life settings, and in that regard if we resample the minority classes it might introduce artificially augmented bias. In the worst case, it might have contaminated the training set with samples that are less likely to occur, which then degrades the accuracy.

5.2.2 Performance Difference in Defect Category with Oversampling

SVM

	precision	recall	f1-score	support
0.00	0.69	0.92	0.79	377
0.33	0.00	0.00	0.00	74
0.67	0.00	0.00	0.00	26
1.00	0.71	0.63	0.67	179
accuracy			0.70	656
macro	0.35	0.39	0.36	656
weighted	0.59	0.70	0.63	656

+ SMOTE

	precision	recall	f1-score	support
0.00	0.72	0.85	0.78	377
0.33	0.31	0.11	0.16	74
0.67	0.11	0.08	0.09	26
1.00	0.66	0.61	0.63	179
accuracy			0.67	656
macro	0.45	0.41	0.41	656
weighted	0.63	0.67	0.64	656

In addition, by examining the performance metrics between without oversampling and SMOTE oversampling on SVM, we can conclude that synthetic oversampling can significantly enhance the performance of minority defect categories (0.33 and 0.67).

5.3 Thresholding vs No Thresholding

	SVM		HC6 (CNN)	
	Origin	Thresholding	Origin	Thresholding
Accuracy	0.70	0.70	0.74	0.72
Precision	0.59	0.71	0.71	0.71
Recall	0.70	0.70	0.74	0.72
F1-Score	0.63	0.65	0.71	0.69

The above result implies that after applying thresholding, the overall classification performance of SVM slightly improves while that of CNN slightly degrades. However, if we consider the metrics for monocrystalline alone, both SVM and CNN classifiers have improved performance, but an opposite tendency is observed for polycrystalline. The performance gap between the two solar wafer types also further enlarged and there could be two reasons for this:

In monocrystalline panels, thresholding enables accurate partition of obvious defect features on plain surfaces. The result can strengthen subtle but important traits that indicate the health of cells.

In the case of polycrystalline solar panels, the situation becomes more complex due to the inherent patterns in these panels. The algorithm may misinterpret these normal patterns as signs of damage or defects when thresholding is applied.

In conclusion, the thresholding methods employed for extracting defect features from polycrystalline solar panels might require further refinement, although it has shown a promising trend in improving the detection accuracy of monocrystalline solar panels. The overall performance can be further improved by researching and developing better thresholding methods with consideration of the specific characteristics of polycrystalline silicon panels.

5.4 Data Augmentation vs No Data Augmentation

From our experiments, data augmentation does not necessarily improve ConvNet classification accuracy (0.74 vs 0.71). In fact, we can observe a significant degrade in performance especially highlighted by the F1-Scores of HC6 (0.71) and HC6 with data augmentation (0.64).

A potential explanation for this phenomenon could be that the network weights are being penalised more frequently during the training process since the image data does not appear the same in each iteration. This can be perceived as a sign that our model is not memorising our dataset, but rather trying to find a pattern. Despite its tendency for better generalisation and less overfitting, it is done at the expense of validation accuracy.

Since the model is trained on more data, it also implies a slower convergence to the process. Our current setup may have limited the training outcome and thus the model appears to be less effective. A potential solution to this problem is to allow a more lenient patience for early-stopping and to allow the model to be trained for much more iterations. A larger raw dataset might also help.

5.5 Performance by Classifier Models

	SGD	KNN	DT	GNB	MNB	BNB	MLP	SVM
Accuracy	0.67	0.66	0.56	0.51	0.38	0.57	0.57	0.70
Precision	0.63	0.63	0.58	0.62	0.61	0.47	0.33	0.59
Recall	0.67	0.66	0.56	0.51	0.38	0.56	0.57	0.70
F1-Score	0.60	0.63	0.57	0.55	0.45	0.43	0.42	0.63

From the above comparative summary of performance among all

general-purpose classifiers, we can observe that SVM performs the best on the given image classification task, and the 3 Naïve Bayes algorithms perform the worst. In addition, given the use of a random train-test-split method, by comparing all the results from different iterations, the performance is consistent on SVM with an overall prediction accuracy exceeding 70%.

This could be due to SVM being able to effectively handle high-dimensional data, such as images. Additionally, SVMs are less prone to overfitting than other algorithms.

5.6 Performance by ConvNet Models

VGG16, in general, outperforms HC6 by a slight difference of 2% (0.76 vs 0.74). However, this is something to be expected since VGG16 is a well-established framework and is thoroughly tested and trained by several large datasets.

However, with ImageNet transfer learning, one might ideally expect VGG16 to perform even better, at least to a level closer to its reported accuracy of 92% in ImageNet classification. Our handcrafted model, HC6, on the other hand, performs roughly as good as VGG16 under a domain-specific context. This somehow implies that the effectiveness of transfer learning can be largely affected by the nature of dataset. VGG16 pre-trained with ImageNet might be best at classifying general-purpose RGB images but our task requires classifying less often seen grayscale EL images which ImageNet might not have related data included for training. That explains why VGG16 with ImageNet does not perform significantly better than HC6 with randomly initialised weights.

Given its much-simplified framework (16 weight layers vs 6 weight layers), HC6 is said to have given a satisfactory performance. The computational time required to complete each training (15 minutes vs 1 hour) also makes it a preferable choice for quick assessment.

5.7 Traditional Classifiers vs Convolutional Networks

The classification accuracy of SVM and VGG16 for mono-crystalline cells differs by only a slight 3% (0.74 vs 0.77) while that for polycrystalline cells differs by a larger gap of 8% (0.67 vs 0.75).

Overall, VGG16 outperforms SVM by about 6% (0.76 vs 0.70), which is in line with S. Deitsch's findings [5]. Although this may imply ConvNets' superiority over traditional classifiers in defect classification, SVM is still a viable choice for quick assessment [5].

5.8 Performance by Combination of Preprocessing Methods

In the classifier experiment pipeline, we have assessed the influence of implementing multiple pre-processing methods on the overall model performance.

SVM

	precision	recall	f1-score	support
0.00	0.69	0.92	0.79	377
0.33	0.00	0.00	0.00	74
0.67	0.00	0.00	0.00	26
1.00	0.71	0.63	0.67	179
accuracy			0.70	656
macro	0.35	0.39	0.36	656
weighted	0.59	0.70	0.63	656

+ SMOTE + Thresholding

	precision	recall	f1-score	support
0.00	0.78	0.66	0.71	377
0.33	0.32	0.42	0.36	74
0.67	0.13	0.27	0.17	26
1.00	0.67	0.68	0.67	179
accuracy			0.62	656
macro	0.47	0.51	0.48	656
weighted	0.67	0.62	0.64	656

The above classification reports indicate a decline in performance after applying thresholding and oversampling to the model pipeline. This could be because after applying thresholding to the samples, there exists a loss of essential traits which might have led to several inaccurate predictions. Oversampling the training set, on the other hand, might have introduced an increased bias to the prediction.

However, when considering the original SVM result versus the result obtained after applying thresholding and oversampling, the combination of methods did help with an improved prediction in minority classes.

6. Conclusion

6.1 Experiment Summary

In this project, we have presented frameworks of classifiers and ConvNets employed for training with the ELPV dataset to classify defective solar module cells from EL imagery. Our experiments highlighted significant performance variations among these models when combined with different pre-processing methods. The best-performing classifier is the SVM pipeline, achieving up to a 70% accuracy without employing oversampling, thresholding, or edge detection techniques. Similarly, VGG16 without any pre-processing stands out in ConvNets, reaching an accuracy as high as 76% in defect categorisation. Overall, VGG16 outperforms SVM by 6%.

For mono modules, both models exhibit strong performance with accuracies over 70% while VGG16 edges slightly ahead by a slight 3%. However, only VGG16 sustains an accuracy of over 70% on poly cells, enabling it to outshine SVM by a margin of 6%.

Other pipelines do not necessarily enhance overall accuracy, but our experiments have shown that the use of thresholding, oversampling and edge detection will slightly enhance the precision and recall for minority classes, especially for defect categories 0.33 and 0.67.

Overall, our models demonstrate satisfactory results in classifying defects in PV cells from EL imagery. While VGG16 shows slightly better performance, it comes with a higher computation demand. SVM is much preferable when it is to be used for training under a small footprint constraint [5]. It is worth noting that our handcrafted HC6 exhibits similar traits of performance to that of VGG16 (i.e., slightly inferior to VGG16, but superior to SVM), making it another practical option for quick assessment.

6.2 Future Improvements

From the findings we have concluded in earlier sections, there are some directions we believe can be further explored to improve the classification performance, which include but are not limited to:

1. Enlarged Sample Size – Given our models are trained and tested with only 2,624 samples, an insufficient amount of data input has contributed to imprecise parameter estimates and a higher risk of overfitting. Therefore, we believe that by enlarging the sample size, our models could yield more precise and consistent predictions, ultimately leading to an improved overall prediction performance.

2. Improved Edge Detection – Given time limitations, we have only implemented the relatively simple Laplacian enhancement as the sole option for edge detection. Mono and poly type solar cells are not differentiated differently which results in a drop in performance. Nevertheless, we anticipate that employing advanced methods like Prewitt and Canny, utilised in similar studies to detect micro-cracks and edge discontinuities in EL images [2], can notably enhance the overall classification performance of defective solar cells.

3. Enhanced Thresholding – Due to time constraints, we are unable to explore more advanced thresholding methods and make refinements to our thresholding configuration. However, based on the literature review, several other thresholding techniques remain for us to explore further. These methods have the potential to greatly enhance both the performance and efficiency of model training.

4. Hyperparameter Tuning – Owing to restrictions in both hardware and time, we are unable to attain an ideal, well-refined parameter combination. If a more rigorous model tuning process is possible, we passionately believe that the classification performance can be improved by another 5 - 10%.

5. Feature Extraction – We did not thoroughly explore the use of feature extraction, e.g., SIFT in our model development. As SIFT is a feature extraction method that reduces the image content to a set of key points used to detect similar patterns in other images, it might be another viable option to further improve our model performance.

7. References

[1] S. Deitsch et al. ELPV dataset, 2023. URL: https://github.com/zae-bayern/elpv-dataset

[2] Y. Zhang et al. Research on Image Defect Detection of Silicon Panel Based on Prewitt and Canny Operator. Frontiers in Physics. 9. 701462. http://dx.doi.org/10.3389/fphy.2021.701462

[3] C. Buerhop et al. A Benchmark for Visual Identification of Defective Solar Cells in Electroluminescence Imagery. European PV Solar Energy Conference and Exhibition (EU PVSEC), 2018. http://dx.doi.org/10.4229/35thEUPVSEC20182018-5CV.3.15

[4] S. Deitsch et al. Segmentation of Photovoltaic Module Cells in Uncalibrated Electroluminescence Images. Machine Vision and Applications, 32(4). http://dx.doi.org/10.1007/s00138-021-01191-9

[5] S. Deitsch et al. Automatic Classification of Defective Photovoltaic Module Cells in Electroluminescence Images. Solar Energy, vol. 185, June 2019, pp. 455-468. https://doi.org/10.1016/j.solener.2019.02.067

[6] U. Otamendi et al. A Scalable Framework for Annotating Photovoltaic Cell Defects in Electroluminescence Images. *ArXiv*. https://doi.org/10.1109/TII.2022.3228680

[7] A. Gholamy et al. Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation. International Journal of Intelligent Technologies & Applied Statistics 11(2) (2018) https://scholarworks.utep.edu/cgi/viewcontent.cgi?article=2202&context=cs_techrep

[8] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

[9] H. Han, W. Wen-Yuan, M. Bing-Huan, "Borderline-SMOTE: A New Over-sampling Method in Imbalanced Data Sets Learning," Advances in intelligent computing, 878–887, 2005.

[10] Sáez, José A. & Krawczyk, Bartosz & Wozniak, Michal. (2016). Analyzing the Oversampling of Different Classes and Types of Examples in Multi-class Imbalanced Datasets. Pattern Recognition. 57. 164–178. 10.1016/j.patcog.2016.03.012.

[11] J. S. Weszka and A. Rosenfeld, "Threshold Evaluation Techniques", IEEE Transactions on Systems, Man, and Cybernetics, vol. smc-8, no. 8, Jan, 1978.

[12] D. Park, "Image Classification using Naïve Bayes classifier", International Journal of Computer Science and Electronics Engineering (IJCSEE), vol. 4, no. 3, pp. 135–139, 2016.

[13] Huang, C., Davis, L. S., and Townshed, J. R. G. 2002. An Assessment of Support Vector Machines for Land Cover Classification. International Journal of Remote Sensing, 23, 725–749.

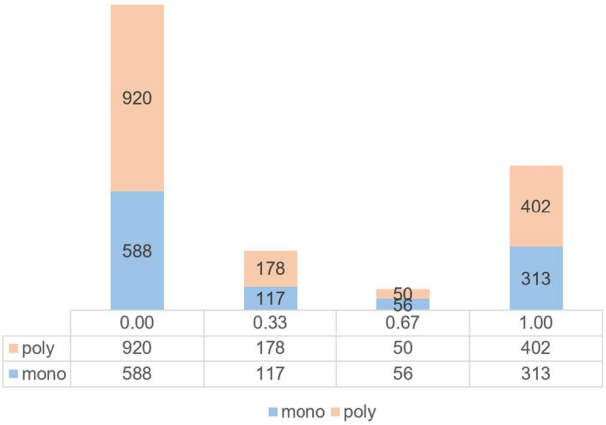
[14] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556v6 [cs.CV], 2015.

[15] UKEssays. (November 2018). Edge Detection Methods in Digital Image Processing.

[16] Jiancheng, Sun, Chongxun, Zheng, Xiaohe, Li, & Yatong, Zhou. (2010). Analysis of the Distance Between Two Classes for Tuning SVM Hyperparameters. Neural Networks, IEEE Transactions on, 21(2), 305-318.

8. Appendix

A. Data Distribution



B. Methodology Overview