Mondragon Unibertsitatea

DOCTORAL THESIS

DEEP LEARNING BASED METHODOLOGY FOR THE DEVELOPMENT OF INDUSTRIAL QUALITY INSPECTION SYSTEMS

JULEN BALZATEGUI ORUNA | Arrasate-Mondragón, 2022

JULEN BALZATEGUI ORUNA | Deep Learning Based Methodology For The Development Of Industrial Quality Inspection Systems

Deep Learning Based Methodology For The Development Of Industrial Quality Inspection Systems

Julen Balzategui Oruna

Supervisors:

Dr. Luka Eciolaza Echeverria

A thesis submitted to Mondragon Unibertsitatea for the degree of Doctor of Philosophy

Department of Electronics and Computer Science

Mondragon Goi Eskola Politeknikoa

Mondragon Unibertsitatea

July 2022

Con cariño, a mis padres y a mi hermano.

Acknowledgments

I would first like to thank my thesis advisor Luka for giving me the opportunity to work on this thesis. Thanks to this thesis, I have been able to delve into the fields of computer vision and artificial intelligence which I have found wonderful fields.

I would also like to thank my colleagues Markel, Iñigo, Javi, Unai, and Oscar for their support, those coffee breaks have helped to make the thesis more bearable.

I cannot but be eternally grateful to Dani Maestro-Watson who has guided me both during the initial days and throughout the three years until the very last day of the thesis. Even when he was busy, he was always able to find a time slot to help me with any of my doubts. Thank you very much for your kindness and your time.

I would also like to thank Mondragon Assembly for providing us with data to carry out the thesis.

Last but not least, all my thanks to my parents and brother. Your help has been indispensable in completing the thesis. You are and will be everything to me. Thank you very much for everything.

Declaration

Hereby I declare that this document is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Julen Balzategui
Arrasate, July 2022

Abstract

In recent years, the manufacturing industry has gone through what has been called the fourth industrial revolution or Industry 4.0. Apart from still automating industrial processes, the revolution has as well brought new trends like zero-defect manufacturing, non-destructive unitary tests, or complete traceability of every part along the production chain. One of the sectors that have been influenced by this revolution is the solar sector. This sector, as part of the strategic sector of renewable energies, has received large funding from government entities and individual investors that have led to an improvement in technology. This has lowered the prices of panels, which in turn has increased the demand for them making it more necessary to automate the production process.

Among all the stages during production, quality control plays a crucial role. In the specific case of the photovoltaic sector, quality control in industrial manufacturing is performed using the Electroluminescence technique which allows practitioners to obtain high resolution images of the photovoltaic cells where defects are highlighted. In contrast to the trend towards automation, in practice, panel inspection is still mostly performed by operators. In recent years, many proposals have been made to automate this quality inspection. However, the proposals made so far show certain limitations for their application in the increasingly dynamic and demanding industrial context.

Some of the identified limitations are: the lack of flexibility to changes in production since the proposed procedures have been designed to take advantage of case specific data features. For example, an inspection system might have been designed to take advantage of the high contrast between the light background and the thin and dark longitudinal cracks in the cells. However, a variation in the data like a darker due to a different material composition of the cells or different shapes of the cracks may suppose to redesign the entire inspection system to adapt to the changes. Other proposals contemplate algorithms that require a large number of representative defective samples for training, which are usually difficult to obtain in industrial environments. And finally, some solutions consist of algorithms that can act as black boxes with respect to their interpretability, which together with giving as a result only

whether a part is defective or not, can raise doubts about the performance of the inspection system.

For these reasons, the objective of the thesis has consisted in designing a methodology based on Deep Learning techniques for the development of inspection systems. The methodology has contemplated techniques that are robust and flexible to changes, but also able to work in industrial environments where there are few available defective samples, and output more interpretable results than a mere classification, for example, the location of defects in the samples. At the same time, the methodology offers ways to obtain inspection models from the very beginning in the production line, and take advantage of their characteristics to obtain more accurate models with almost no need for human intervention.

Resumen

En los últimos años la industria manufacturera ha estado envuelta en lo que ha denominado como cuarta revolución industrial o Industry 4.0. Además de perseverar en la automatización de los procesos, la revolución ha traído consigo nuevas tendencias para producción tales como la fabricación sin defectos, un control de calidad no destructivo unitario, o el rastreo absoluto de las piezas a lo largo de la cadena de producción. Entre los distintos sectores influenciados por la revolución, se encuentra el sector fotovoltaico. Este sector, ha recibido gran financiación de entidades gubernamentales e inversores privados que ha derivado en una mejora de la tecnología. Esto ha hecho que los precios de los paneles se hayan abaratado, aumentando así la demanda de los mismos, haciendo a su vez más necesaria la automatización de su proceso de producción.

Entre todas las etapas durante la producción, el control de calidad juega un papel de vital importancia. En el caso concreto del sector fotovoltaico, el control de calidad en su fabricación industrial se realiza valiéndose de la técnica de Electroluminiscencia, la cual que permite obtener imágenes de alta resolución de las células fotovoltaicas donde los defectos quedan resaltados. En contraste con la tendencia hacia la automatización, en la práctica la inspección de los paneles sigue realizándose mayormente por operarios. En los últimos años numerosas propuestas han sido realizadas con el objetivo de automatizar este control de calidad. No obstante, las propuestas hasta el momento muestran ciertas limitaciones para su aplicación en el contexto industrial cada vez más dinámico y demandante.

Entre las limitaciones identificadas se encuentran: la falta de flexibilidad a cambios en la producción ya que los procedimientos propuestos han sido diseñados para sacar partido de particularidades muy específicas de los datos. Por ejemplo, el sistema de inspección puede haberse diseñado teniendo en cuenta el gran contraste que el fondo claro de la célula y las grietas negras y longitudinales en las mismas presentan. No obstante, una variación, como por ejemplo, un fondo más oscuro debido a una nueva composición de las células o grietas con distinta morfología, puede suponer la necesidad de tener que volver a diseñar el sistema de inspección por completo. Por otra parte, algunas propuestas contemplan algoritmos que requieren muchas muestras

defectuosas para su entrenamiento, las cuales suelen ser de difícil acceso en entornos industriales. Y por último, algunas soluciones consisten en algoritmos que pueden actuar como cajas negras respecto a su interpretabilidad, que en conjunto con dar como resultado solo si una pieza es defectuosa o no, puede suscitar dudas sobre el funcionamiento del sistema de inspección.

Por estas razones, el objetivo de esta tesis ha consistido el diseño de una metodología basada en técnicas Deep Learning para el desarrollo de sistemas de inspección. La metodología ha contemplado técnicas robustas y flexibles a cambios, pero capaces de funcionar en entornos industriales con escasas muestras defectuosas, y además ofrecer resultados más interpretables que una mera clasificación, como por ejemplo, la localización de los defectos en las muestras. A su vez, se ofrecen maneras de obtener modelos de inspección desde un primer momento en la línea de producción, y aprovechar las características de los mismos para obtener cada vez modelos más precisos sin casi necesitar una intervención humana.

Laburpena

Azken urte hauetan fabrikazio industriala bete betean sartuta egon da industry 4.0 edo laugarren industriala-iraultza deitu den prozesuan. Iraultza hau, aurretik prozesu industrialen automatizazioaren jarraipena mantenduz gain, beste tendentzi berri batzuk ere ekarri ditu, esate baterako, akatsik gabeko produkzioa, kalitate kontrol ez intrusibo unitarioa, edo fabrikazioan zehar pieza guztien kontrola eramatea. Iraultza hau pairatu duten sektore ezberdinen artean, panel fotovoltaikoen sektorea dago. Sektore hau, energia berriztagarrien sektore estrategikoaren barruan izanda, gobernuen eta inbertsore pribatuen finantzaketa handia jaso du gaur egungo teknologia hobetzeko helburuarekin. Honek, panelen prezioa behera egitea eragin du, ondorioz panelen eskaera handituz eta automatizazioaren beharra ere areagotuz.

Fabrikazio prozesuaren etapa guztien artean, kalitate kontrola paper oso garrantzitsua betetzen du. Sektore fotovoltaikoaren kasu konkretuan, kalitate kontrola Elektroluminsentzia deituriko teknikaren bitartez egiten ohi da, non paneletako defektuak nabarmenduta agertzen diren erresoluzio altuko irudiak ateratzen dira. Nahiz eta prozesuen automatizaziorantz mugitu, gaur egun sektore honetan langileek jarraitzen dute izaten kalitate kontrolaren egiten dutenak. Horregatik, azken urte hauetan hainbat proposamen egin dira automatizaziorantz begira. Hala ere, proposaturiko soluzioak hainbat limitazio azaltzen dute kontextu industrial batean aplikatu ahal izateko.

Limitazioen artean hauek nabarmentzen dira: kanpo aldaketei aurre egiteko malgutasun gutxi dute zeren aplikaturiko prozedimenduak datuen ezaugarri oso partikularrak kontuak izanda diseinatu dira. Esate baterako, inspekzio sistema diseinatzerako orduan, zelularen kolore argia eta defektu ilun eta longitudinalaren arteko kontraste handia aprobetxatu da. Baina zelularen material konposizio berri baten ondorioz zelula aspektu ilunago bat izaten badu edota defektuen itxura aldatzen bada, posible da soluzio osoaren berdiseinuaren beharra izatea. Beste alde batetik, soluzio batzuk entrenamendurako akastun lagin asko behar duten algoritmoak erabiltzen dute. Halako laginak ezin dira beti erraz lortu, eta kontextu industrial batean are eta zailago izan daiteke. Eta azkenik, erabilitako algoritmo batzuk kaxa beltz bat izango lirateke moduan joka dezake, non pieza bat txarra edo ona

den bakarrik iragarriz erabiltzaileen aldetik momentu batzuetan sistema ondo funtzionatzen duen ala ez inguruan mesfidantza sortu dezake.

Arrazoi hauengatik, tesi honen helburu nagusia Deep Learning oinarriturik kalitate kontrolerako sistema bat eratzeko balioko duen metodologia bat eratzea izan da. Metodologiak teknika sendo eta flexibleak jorratu ditu, beti ere akastun diren datuen behar minimoa kontuan izanda eta emaitz interpretableak emateko kapazitatea bermatuz, hala-nola akatsen posizioa emanez. Honez gain, metodologia baita lehenengo momentutik inzpekziorako modelo bat lortzeko aukera eta gero gizakiaren parte hartze minimoarekin modeloak hobetzeko aukera ematen du.

Table of contents

List of figures

List of tables

List of abbreviations

AE AutoEncoder

ANN Artificial Neural Network

AUC Area Under Curve

BN Batch Normalization

ConvL Convolutional Layer

CNN Convolutional Neural Network

DL Deep Learning

EM Earth-Mover's Distance

EL Electroluminescence

FC Fully Connected layer

FCN Fully Convolutional Network

FN False Negative

FP False Positive

GAN Generative Adversarial Network

GD Gradient Descent

IoU Intersection Over Union

IR Thermography

ICA Independent Component Analysis

ICTs Information and Communication Technologies

JS jensen-shannon divergence

MaxPool Max Pooling

ML Machine Learning

MLP MultiLayer Perceptron

MSE Mean Square Error

NMAP Normalized Masked Average Pooling

PRO Per Region Proposal

PV Photovoltaic

ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent

sPRO Saturated Per Region Proposal

SVM Support Vector Machine

TL Transfer Learning

TN True Negative

TP True Positive

WGAN Wasserstein Generative Adversarial Network

WGAN-gp Wasserstein Generative Adversarial Network with Gradient Penalty

Chapter 1

Foundation and Context

This first chapter is going to briefly describe how industrial production has evolved in the last years, and how the evolution has brought to light some limitations of the techniques that have been applied to the different proposed solutions. Particularly is going to focus on the renewable energies sector and the quality inspection process within the whole production process which have served us to define the context of the thesis. Also, it will outline the main objective to achieve with this thesis, as well as, enumerate the publications as a result of the works done in the way to meet such objective.

1.1 Motivation and scope of research

Since the third industrial revolution, industrial production processes began to be automated through the inclusion of Information and Communication Technologies (ICTs) into the existing processes, allowing companies to reach higher levels of automation and also productivity and efficiency. This trend has continued over the years up to the present day. These days, the industry is involved in the so-called fourth industrial revolution or Industry 4.0, in which the use of sensors throughout the production processes has been accentuated. These sensors generate a large amount of data related to the process, causing the rise of the application of new techniques, mainly from the Artificial Intelligence field, which are based on the use of such data in order to extract valuable information from them to improve these processes.

Along with the surge of these techniques, Industry 4.0 has also made the way to new trends such as the customization of the production for each client's requirements. This can be translated as the need for flexible and robust processes that can rapidly adapt to all sorts of forthcoming changes without supposing a detriment in the final product quality. In certain sectors, for example, the automotive or aerospace sector, the quality inspection process is required to be very meticulous, as a faulty sample can mean a non-compliance with the signed contract and result in a big monetary loss for the manufacturing company. Moreover, nowadays industrial production is moving towards zero-defect manufacturing standards, with unitary nondestructive inspection procedures, and with a complete traceability of produced parts, making the quality inspection even more important within the whole production process.

Among the different industrial sectors, international investors and global governments have been attracted to the renewable energy sector. This sector has proved as an alternative source of energy that can serve as a clean substitution or complementary energy source to traditional energy sources such as gas, coil, or oil. These clean energy sources have important advantages over traditional ones, for example, they do not present a limitation with regard to the "material" from where the energy is extracted. Or also, the energy extracted from renewable sources is more environmentally friendly during its use. These are extremely important characteristics in the current global situation, as there are signs that we are running out of fossil fuels reserves and humanity is continuously demanding more energy. The advantages that present renewable energies have attracted a great amount of economic investment in the last years (nearly 1.3 trillion dollars into solar energy illustrated in Figure 1.1).

Global power sector investment is set to increase by around 5% in 2021 to more than 820 billion dollars, its highest ever level, after staying flat in 2020. Renewable energies are dominating investment in new power generation capacity and are expected to account for 70% of the total this year. And that money now goes further than ever in financing clean electricity, with a dollar spent on solar Photovoltaic (PV) deployment today resulting in four times more electricity than ten years ago, thanks to greatly improved technology and falling costs [50].

In addition, the incursion of new companies in the sector has increased the competitiveness of the market, making the price of PV panels drop at a rapid pace [106]. As the trend has been accentuating in the last years, the demand for the installation of such panels has also increased by 36,8% from 2010 to 2018. It is expected that world governments are going to keep betting on renewable energies as a strategic sector of the economy

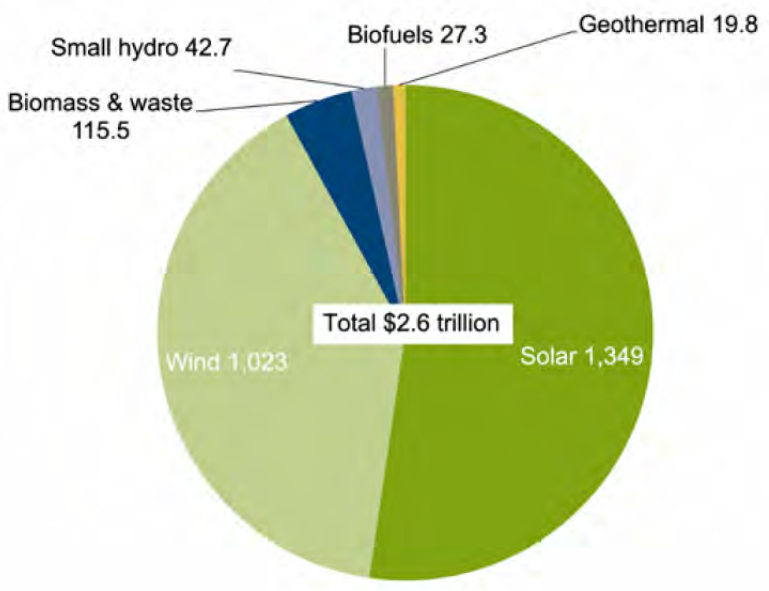


Figure 1.1: Renewable energy investment capacity over decade, 2010-2019. Source: [17]

aiming to reach net-zero emissions by 2050, although the consequences of the Covid-19 pandemic and the recent war situation in Ukraine might impact regarding the timing of the execution of the plan.

In concordance with the increase in the market demand for PV panels, manufacturing companies have also intensified their production of panels [52]. This supposes that in the whole production process, there will be a greater number of PV modules [1] that will need to be handled as well as an increasing rhythm of the production. Because of this, big efforts have been lately done to automate the manufacturing process. The goal is to enhance production capabilities by being able of working with larger module quantities, under more strict production times, and at lower costs. Thus, incorporating nondestructive unitary testing is a must in order to improve the process control, enabling to reach 100% of traceability of the produced parts.

In the case of solar panel production, the imaging technique called Electroluminescence (EL) is commonly employed during the quality inspection stage. The procedure in this technique consists in putting the panels under an electrical current flow causing the cells to emit light by the effect of the phenomenon called Electroluminescence [40]. The light is then captured in high-resolution grayscale images that represent how well the electricity can flow from the different areas in the cell. The areas that present proper levels of conductivity will appear lighter, while the areas with lower levels of conductivity

[1] *PV cells* and *PV module* are used across the document interchangeably

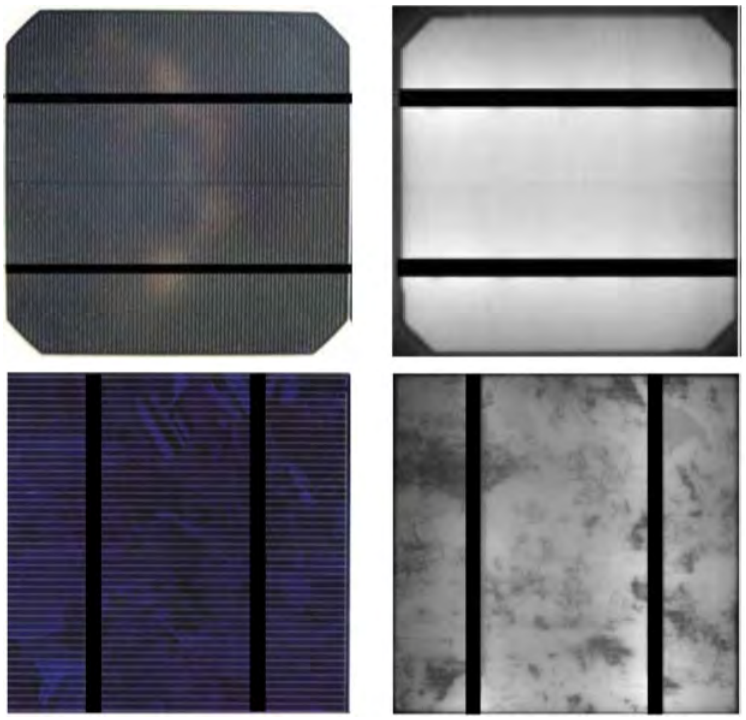


Figure 1.2: Electroluminescence image examples of polycrystalline solar cell (top) and monocrystalline solar cell (bottom) looked by the naked eye (left) and in El image (right). Image source [40]

will appear darker. Figure 1.2 shows what a cell would look like if it is seen with the naked eye, and on the left how the same type of cell would look like after employing the Electroluminescence technique. If the cells are in good condition, just the areas that are not supposed to conduct electricity will remain dark. However, if the cell has been under excessive mechanical stress during the module transportation, has not been correctly soldered or contains any kind of failure, the affected areas will end up not conducting the electricity, and thus, appear dark easing their detection. This imaging technique helps during the inspection as it is not easy to identify defects in cells with the naked eye.

Nonetheless, the availability of these kinds of images does not prevent the companies from performing a manual inspection where human operators need to check EL images of each cell in the panel in order to detect possible defective cases. Usually, the time to inspect the entire panel that is on average composed of 60 cells is stipulated to be 30 seconds, which means more or less half a second per cell. The human operator might struggle to keep such production pace for a long period of time. Also, there is an inherent evaluation subjectivity that each operator has which will be hardly shared among other relay mates, and can even change for the same operator from day to day based. All these

factors can lead to irregular inspection criteria and unpredictable outcomes which is not desirable when the objective is to reach high quality panels.

In recent years, this task has been subject of automation within the present industrial automation trend. Several proposals have been made towards this end, from more purely traditional manual feature engineering based procedures, to a mix between a manual engineering approach and shallow Machine Learning (ML) methods, and to more recent Deep Learning (DL) based approaches. Manual feature engineering based procedures usually reach high defect detection rates while demanding minimum resources for their applications. However, they usually consist of very case specific solutions that present a lack of adaptability, and thus could require a complete redesign of the system if a change in the inspection is required. Taking into account nowadays constantly changing environment, these kinds of proposals can suffer to keep up with such changes. Because of that, the latest proposals have moved toward Deep Learning methods as they have shown capable of automatically extracting meaningful features directly from raw data. In this way, they present higher levels of adaptability as well as a high level of defect detection capability making them more interesting for the current dynamic scenarios. This thesis has explored these latter methods in the context of the quality inspection within the production of solar cell panels.

1.2 Objectives and Contributions

Many initiatives/attempts are targeting the Automatic Quality inspection of solar panels. However, the proposed solutions to this day are not as generalizable to different scenarios as one might wish as they heavily rely on case and situation specific features in order to design the inspection system. This can result in the need to allocate a considerable amount of resources when the need to adapt the system to changes in production arises. This might not be affordable for all companies, or not worth the effort, thus nowadays the inspection is still lacking automation.

Because of this, the objective that was established for this thesis consists in proposing a DL based methodology for the development of a robust and flexible industrial inspection system. All the experiments were carried out within the context of the inspection of solar modules, however, it should be extensible for other domains. Moreover, the techniques that were selected during the design of the methodology are relative to Deep Learning methods for image segmentation that uses few defective data for training which were

chosen after reviewing the state of the art in quality inspection of solar panel production in Chapter 3. The proposed methodology is illustrated in Figure 1.3 which is composed of three stages that tackle different periods that an inspection system can face during its cycle of life:

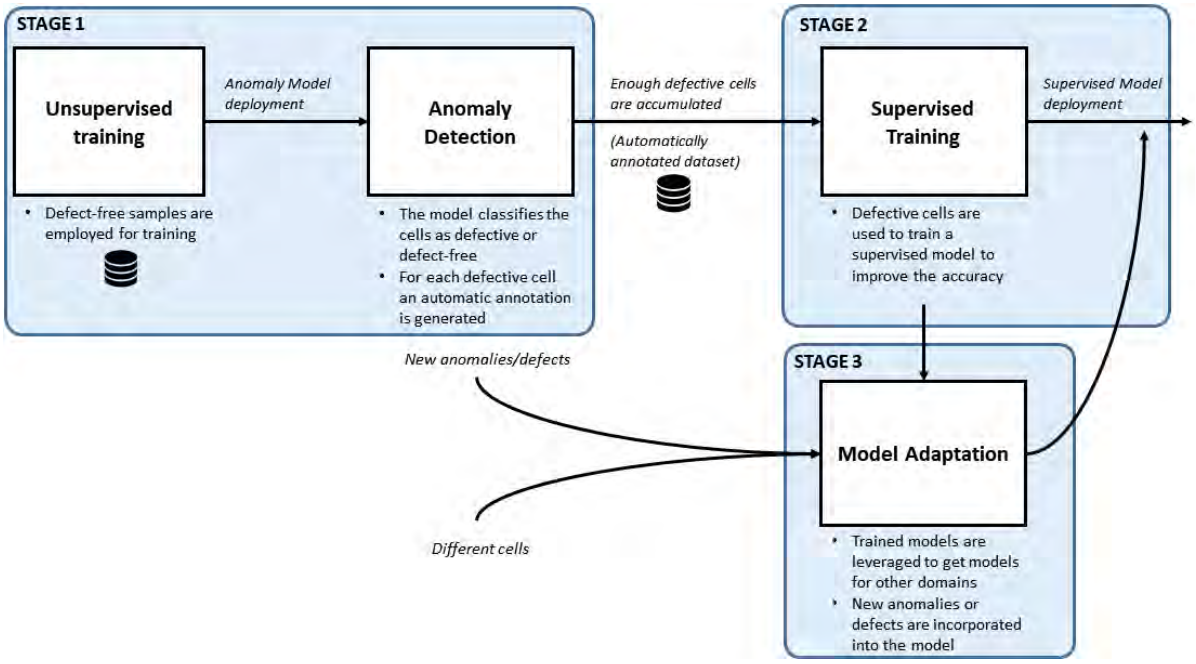


Figure 1.3: Overall schema of the proposed methodology for the development of an inspection system.

- **Stage 1 - Anomaly detection:** In the first stage of the set-up of a new production line, an anomaly detection approach would be used. By training a network using only non-defective cells, the cells that are out of normality, i.e. anomalous or defective, would be identified. In addition to classifying them, the defective areas within the cells would also be marked for inspection, i.e. defect segmentation. In this way, it would not be necessary to wait for having enough defective cells for training, and it would be possible to start detecting defects from the beginning.

- **Stage 2 - Supervised training:** During the lifetime of the production line, it will generate defective cells which will be automatically identified and annotated at pixel level by the model from the first stage. Once a handful of these defective samples are accumulated, they will be used to train a second model in a supervised manner and using methods that require little data for training. This second model will be trained specifically to search for concrete defects reaching higher accuracy rates than the previous model in detection.

- **Stage 3 - Model adaptation:** In a production system, there may be features that are not common and rarely appear during production. It could also happen, that similar inspection scenarios could take advantage of existing models. For this case, the methodology proposes two DL bases techniques called Few-shot learning and Transfer Learning, which take advantage of the previously trained models to obtain new models that will be adapted to work on the new line. Using already trained models as the starting point alleviates the need for defective data which accelerates the deployment of the new line.

1.3 Publications

The design of the methodology required to experiment with several techniques in order to check their applicability. These experiments lead to different publications that are enumerated below:

First Author publications

- [8] Balzategui, J., Eciolaza, L., Arana-Arexolaleiba, N., Altube, J., Aguerre, J.,Legarda-Ereño, I., Apraiz, A.: Semi-automatic quality inspection of solar cell basedon convolutional neural networks. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 529–535. Zaragoza, Spain (2019). DOI 10.1109/ETFA.2019.8869359.

- [9] Balzategui, J., Eciolaza, L., Arana-Arexolaleiba, N.: Defect detection on polycrystalline solar cells using electroluminescence and fully convolutional neural networks.In: 2020 IEEE/SICE International Symposium on System Integration (SII), pp.949–953. IEEE, Honolulu, HI, USA (2020). DOI 10.1109/SII46433.2020.9026211 .

- [10] Balzategui, J., Eciolaza, L., Maestro-Watson, D.: Anomaly detection and automatic labeling for solar cell quality inspection based on generative adversarial network. Sensors21(13) (2021). DOI 10.3390/s2113436.

- [7] Balzategui, Julen, and Luka Eciolaza. "Few-shot incremental learning in the context of solar cell quality inspection." arXiv preprint arXiv:2207.00693 (2022). (Submitted).

Collaborations

- [70] Maestro-Watson, D., Balzategui, J., Eciolaza, L., Arana-Arexolaleiba, N. (2020). Deflectometric data segmentation for surface inspection: a fully convolutional neural network approach. Journal of Electronic Imaging, 29(4), 041007.
- [69] Maestro-Watson, D., Balzategui, J., Eciolaza, L., Arana-Arexolaleiba, N. (2019, July). Deflectometric data segmentation based on fully convolutional neural networks. In Fourteenth International Conference on Quality Control by Artificial Vision (Vol. 11172, p. 1117209). International Society for Optics and Photonics.

1.4 Outline

The rest of the document is organized as follows:

- **Background**: The first chapter describes the background of the thesis in terms of the basics of the techniques that have been employed in the thesis, an outline of the use case that has been used to perform the experiments, and the metrics, data, and hardware and software specifications that have been used in the experiments to train the techniques and evaluate their results.
- **Literature review**: the second chapter provides a literature review that revolves around the techniques that has been applied so far in the context of the thesis.
- **Methodology (chapter 7)**: the following chapters, **Supervised learning (chapter 4)**, **Anomaly detection (chapter 5)**, and **Model adaptation (chapter 6)** describe the three stages that compose the proposed methodology as well as the experiments that were performed to check their applicability as techniques in each stage. Also a real experiment is described on how the methodology would be applied in a real scenario.
- **Conclusions and future works**: The last chapter concludes the document with some conclusions that were drawn during the experimentation in the thesis, and also some suggestions for future work lines that can be followed.

Chapter 2

Background

The thesis has mainly focused on the use of Deep Learning methods to design the methodology for automatic solar cell quality inspection. This section will present a brief background about the different terms used throughout the next chapters in the thesis aiming to give some context that can help to understand them.

The background will be divided into two parts: first, some general concepts about how DL methods work and how they are evaluated will be introduced. Then, more of this thesis specific concepts will be described which will be related to the solar panel production process, and the data, hardware and software used in the different experiments.

2.1 Deep Learning

Deep Learning is a subfield within the Machine Learning field that is at the same time a set of methods that are part of the broad domain of Artificial Intelligence (AI). The aim of Artificial Intelligence methods nowadays is to try to automate the decision-making process by means of employing models trained on data gathered from the process, rules defined by experts, or mathematical functions that describe how the process works.

Among the various alternatives proposed, ML methods have chosen to use data as the backbone of their strategy to achieve the goal of making machines intelligent. Broadly speaking, ML methods try to train algorithms to learn to extract meaningful features from data, thus, later these features can help the algorithm perform optimally in a given

task such as data classification, regression, or clustering. When talking about training, it often referred to feeding data samples to the algorithms in an iterative way so that they can process them and make a prediction or estimation for each sample. Once this prediction has been made, it is evaluated by means of an error function that serves to guide the algorithm in the following iterations towards a specific objective, for example, to learn to classify images. The form of evaluation and the objective is usually dependent on the nature of the data as well as the objective set, which is further elaborated in the next section.

Over the years, different ML algorithms have been proposed (e.g. Support Vector Machine (SVM), K-means, linear regression...etc) and successfully applied in many different fields. However, in recent years, as a result of having easier access to more data and more computing power, the field has seen the resurgence of a new type of methods that compose a subfield within ML known as Deep Learning. The methods within this subfield are characterized by the use of artificial neural networks as their core technique, which like the aforementioned ML methods, are iteratively trained to extract features from the data and then apply them to a defined task. Nonetheless, these latter methods' training is a more data intense process. In contrast with ML methods, neural networks usually require thousands of data samples in order to obtain robust models than can generalize well cases outside of the training set. Despite such limitations, since their emergence neural networks have demonstrated great accuracy in various tasks such as image classification or text classification, which has displaced more traditional techniques to a less relevant position becoming the way to go in fields like Computer Vision or Natural Language Processing.

2.1.1 Types of learning

When talking about the training of a neural network, it refers to optimizing the neural network with a given data such that it can then be employed to solve a task, for instance, an image classification problem. However, the type of data available for training is not always the same. The type of the available data will determine which type of training can be carried out, which could be categorized into four types of learning: supervised learning, unsupervised learning, semi-supervised, or reinforcement learning [44].

- In **Supervised learning**, the algorithms are trained using a dataset with form $D = \{\{x_1, y_1\}, ..., \{x_n, y_n\}\}$, where for each of the samples x, there is an output y that is known beforehand. Based on the pairs of input and output, the algorithms

are trained such that, for each input, it gives its correspondent or expected output. This is achieved by making the network minimize a given loss function $L = (x, f(x))$ which will try to force the network to behave as expected to every input data. This kind of training is widely used in classification kind of problems, where the algorithms are trained to classify the data points into predefined output classes.

- In **unsupervised learning**, the dataset samples do not have an expected output (label) associated as in supervised learning. Due to the lack of labels, an exploratory search is conducted in order to find similar features or patterns among the data sample such that they can be then clustered together. In the context of deep learning, for example, unsupervised learning is commonly applied in anomaly detection problems, where using only normal data as training data, the networks are forced to learn the probability distribution of that training samples, P_X, so then can be used to detect anomalous samples. This type of learning, however, does not reach the same accuracy rates as the supervised type of learning due to the exploratory nature of the training itself. While in one a specific objective is set and the network is trained accordingly, in the other possible patterns and structures are searched in the data which usually leads to more noisy outcomes.

- **Semi-supervised learning** is a mix of supervised and unsupervised learning. In this case, the network is trained with a dataset composed of labeled and unlabeled data points. While the training can be closer to unsupervised learning, in this case, labeled samples are incorporated to help the network during the feature extraction process.

- In **Reinforcement learning** the network does not learn from the error gradient calculated from the network prediction and the expected value but from rewards and penalties received during "experimentation" with the environment. By interacting (actions a) with the environment, the model passes from one state s to another resulting in different rewards r. In the end, the network is trained to maximize the final return given by the sum of the rewards, $G = \sum_{t=1}^{\infty} R_t$.

These learning approaches can be used in different industrial scenarios, for example, for defect classification (supervised training) [1, 68, 105, 61], anomaly detection (unsupervised training) [89, 46] or robot picking (reinforcement learning) [34, 114].

2.1.2 Architectures

In Deep Learning, the models are neural networks: a conjunction of interconnected neurons that define mathematical functions that map some set of input values to an output value $y = f(x; \theta)$, where θ is the parameters in the network. The mapping, illustrated in Figure 2.1, consist in performing a dot product between the data coming from input connections and the weights associated to each connections, plus a bias term, $y = \sum_i W_i \cdot x_i + b$.

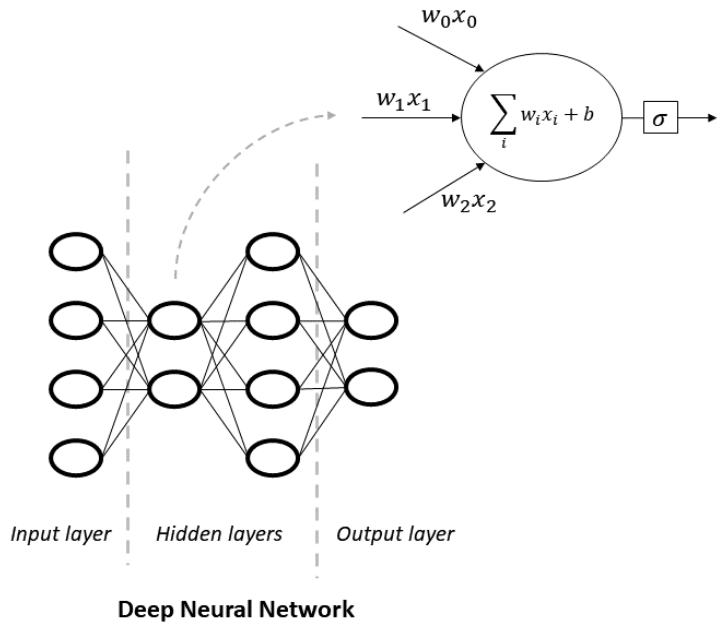


Figure 2.1: Deep Neural Network architecture schema

The neurons are organized in layers stacked one after the other, and the connections are established between adjacent layers of neurons as illustrated in Figure 2.1. However, stacking linear functions one after the other can only lead to representing more and more complex linear functions at each time. For instance, a neural network composed of three layers defining three functions, f^1, $f^2$and f^3, will compose a chain $y = f(x) = f^3(f^2(f^1(x)))$ that in the end still be a representation of a linear function.

In order to be able to approximate more complex functions that will serve to represent nonlinear data such as images, the design of a neural network usually incorporates a nonlinear activation function σ, at the end of each neuron, $y = \sum \sigma(W_i \cdot x_i + b)$. Some of the most common nonlinear functions used when constructing neural networks are Sigmoid, Tanh, ReLU, or LeakyReLU. With these functions, the output of the neurons is forced to fall into different data ranges as can be observed in Figure 2.2 which breaks

down the linearity of the models. The choice among the existing functions will depend entirely on each practitioner, as every case could be different. To give an example, if we want the network to output the data in a range $[0, 1]$ so we could train it as if it was a probability output we could choose the Sigmoid function to be the final activation function of the network.

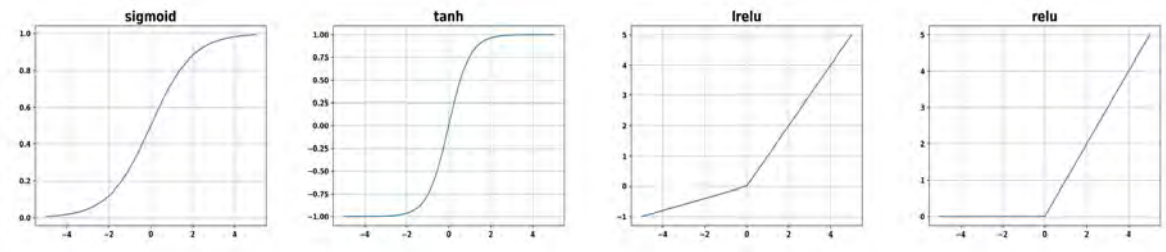


Figure 2.2: Some of the most common activation functions used in neural network.

The layers are denominated as "visible" or "hidden" based on accessibility to the inputs and outputs. The input and output layers are the visible layers in the network as the first is where the data enters the network and the latter where the data get out of the network. Then, all the layers in between these two are considered the hidden layers of the network.

As can be observed in Figure 2.1, every input connection has its own associated weight value. The value of the weights will determine how the information will "flow" through the network. The optimization of the networks aims to find the best combination of the weight values, such that the network will yield the best performance in the given task (e.g. higher accuracy at a classification task).

Different types of architectures have been proposed throughout the years in order to deal with different kinds of problems, as well as try to solve some of the drawbacks of older models. Nowadays, in problems where image processing is somehow involved, the most widely employed neural network architecture is the Convolutional Neural Networks (CNN) illustrated in Figure 2.3. CNNs are a type of deep neural network that assumes the input data to have a three-dimensional shape (i.e., images). While in the initial architectures (i.e, Multilayer Perceptron (MLP)) the neurons were fully connected between layers, in CNNs, neurons are connected to a small region in the input data, thus the weights in that region (i.e., *receptive field*) are spatially shared. In this way, the spatial relationship between the pixels in the data will remain as it could contain meaningful information. In addition, if a MLP was applied on such high dimensional data, the memory usage would increase notoriously as the number of neurons needed to process the data would be huge.

In the case of CNN architecture instead, the images are processed using a receptive field which reduces the number of neurons needed to extract the features, and in this way alleviate the hardware demand making them suitable to work with image type of data [102].

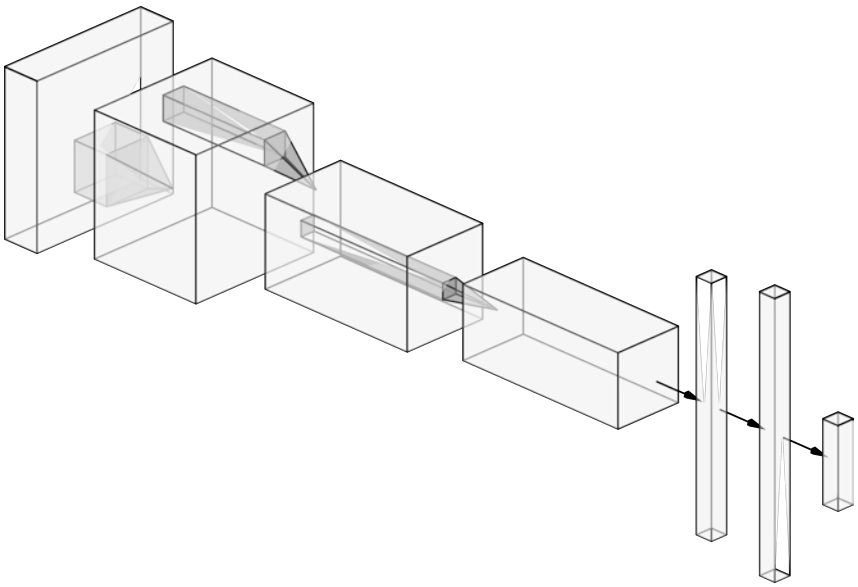


Figure 2.3: Convolutional Neural Network architecture schema. Source: http://alexlenail.me/NN-SVG/AlexNet.html

The principal characteristic that makes CNNs different from other types of networks is the Convolutional layers, which have parameters consisting of a set of learnable filters. These filters are convolved (hence the name of the layer) across the height and width of the input volume computing the dot product between the entries of the filter and the regions in the input volume (i.e., the previously mentioned receptive field). The convolution will be performed for each of the filters defined as a hyperparameter in the layer, yielding 2d activation maps that will correspond to the responses of the input volume to each of the filters.

Conceptually, these filters will end up being akin to the types of filters that traditionally users defined in conventional computer vision pipelines, for example, blur the images, extract edges, find circles, or find corners. However, in contrast to traditional manual labor, the filters in Deep Learning are automatically defined by the network throughout a training. During the training, the network will decide on its own how to configure the filters at each layer such that it can extract meaningful features from the data, and thus optimize its overall performance for the given task (e.g., classification).

In addition to the Convolutional layers, other types of layers are also usually employed in the CNNs architectures. To name a few, pooling layers like Max-Pool or Avg-Pool are usually incorporated to progressively reduce the spatial size of the input volume to reduce the number of parameters in the networks. Or, the Batch Normalization layer (BN) that normalizes the output activations after each Convolutional layer, which has been shown as a way to speed up the training process[51]. Or also, Dropout layers that will deactivate certain neurons in the layers during the forward pass as a form of regularization that will prevent the network from overfitting the data (i.e., memorizing the training data). For a more extended explanation and visual animations of how the layers work refer to Stanford's lecture on Convolutional Neural Networks.

The first publication of a successful application of CNN was done by [62] aiming to identify different zip codes, digits, ...etc from images. The network was called LetNet-5 and was composed of a sequence of three 3 layers: convolution layers, pooling layers, non-linearity function, and fully connected layers with a final activation function at the end that acted as a classifier.

Nevertheless, the turning point in the usage of the CNNs was their application in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012, where [60] proposed AlexNet and surpassed the accuracy results of all other contestants that were employing more traditional image processing based approaches. From that moment on, CNNs were adopted by the Machine Learning community as one of the *de facto* techniques to be used for modeling computer vision problems. Also, in recent years, this architecture has evolved into more complex structures to adapt to different tasks, such as ResNet [47] or Inception [93] for image classification, YOLO [80] or Mask R-CNN [48] for object detection, or Fully Convolutional Networks (FCN) [65, 81] for image segmentation.

2.1.3 Optimization

Apart from the architecture configuration, the loss function is another key component during the training process. The loss function, Equation 2.1, serves as the final evaluation of the network that will measure the magnitude of the error when predicting an output for the given input x. In other words, it will measure how far the prediction was for a given sample regarding the expected output y. The error function also implicitly evaluates how optimum the parameters of the networks θ at that training stage.

$$\mathcal{J}(\theta) = \mathcal{J}(\sigma(x;\theta), y) \tag{2.1}$$

In the end, the training of the neural network is an optimization problem where the objective is to find the global minima of the error function (or at least a local minima), meaning that the network has learned to do the given task for the given data. Neural networks are usually trained with Gradient Descent (GD) [16] and Back-propagation algorithms [83] which are focused on updating all the weights based on searching the direction of the greatest increase rate of the loss function. As the objective is to reduce the overall error, and this is parameterized by the weights in the network, the weights are updated based on the opposite direction of the gradient (gradient descent) with respect to the error which can be expressed as in Equation 2.2.

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{J} \tag{2.2}$$

where θ are the parameters in the network, α is the magnitude of the update known as the *learning rate*. The update is done at every training iteration, which consists of a forward and backward pass through the network. In the forward pass, first, the data goes through the layers in the network where the features are extracted. Then, the extracted features are employed to complete the given task, for example, image classification. After the network predicts an output, an error is calculated using a loss function that measures the distance between the output and the expected value. Once the error is computed, this is backpropagated from the final layer to the very first layer in the network such that, each weight can be updated using the gradient to respect the error expecting to reduce the overall error of the network each iteration. In this way, the global minima will be steadily approached iteration after iteration until, hopefully, the global minima is reached or the training is stopped.

This process is done using the Back-propagation algorithm that can efficiently compute the gradient of the loss function w.r.t. all the parameters in the model. The algorithm takes advantage of the fact that the neural networks are function compositions and applies the chain-rule to compute the gradients with respect to each parameter. The chain-rule says that the derivative of composed function $F(x) = f(g(x))$ w.r.t x is the derivative of the outside function times the derivative of the function inside, i.e. $= F'(x) = f'(g(x))g'(x)$. Following this rule, it is possible to compute the error with respect to the parameters at the last layer in the network and by means of the partial derivatives and the function composition structure defined by the network architecture, propagate

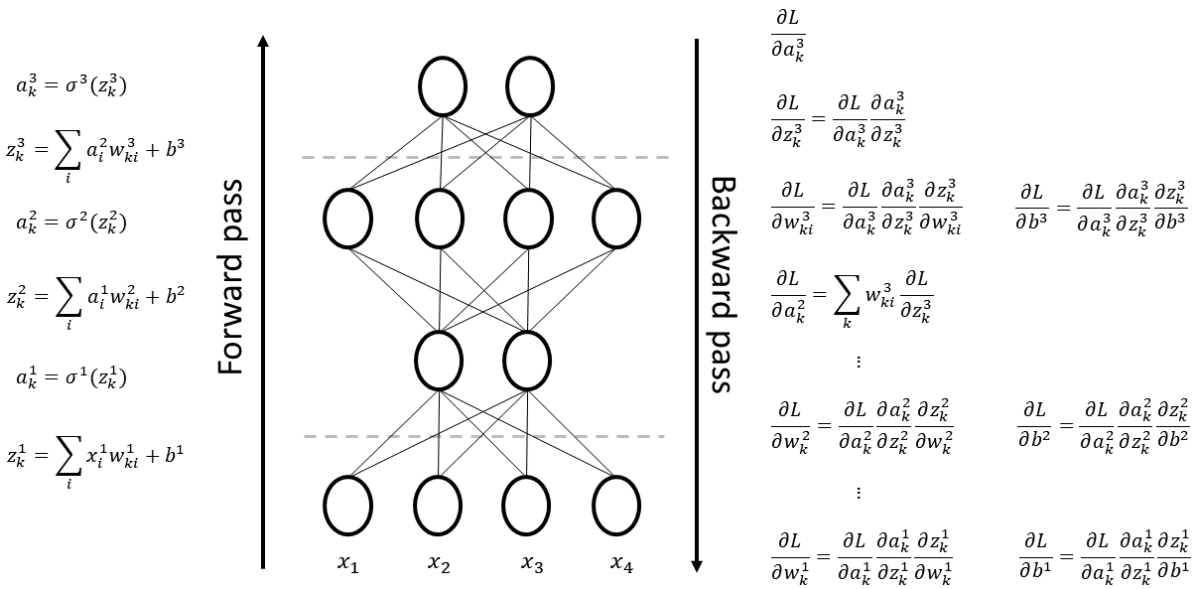


Figure 2.4: Forward pass and error backpropagation example. The superscript denotes the layer, k denotes the neuron at the layer, and i the connection to the neuron.

the error back to the initial layers i.e. backpropagate the error. Figure 2.4 illustrates this forward and backward process. For example, to compute the gradient of the error w.r.t the weight w_{ki}^3, i.e. that connects the first neuron in the last layer and the first neuron in the previous layer (i.e. $k = 1$ and $i = 1$), the relation of this weight to the error must be taken into account which is defined by the expression $\sigma(\sum_i a_i^2 w_{ki}^3 + b^3)$, where a is the output coming from the previous layer that acts as input, w the weight associated to the input and b the bias term. If the functions in the expression are denominated as follows, $z_k^3 = \sum_i a_i^2 w_{ki}^3 + b^3$, and $a_k^3 = \sigma(z_k^3)$, the chain-rule will define that partial derivative of the error with respect to the weight w_{ki}^3 should be defined by taking the partial derivatives of the outside function times the function inside, i.e. $\frac{\partial L}{\partial w_{ki}^3} = \frac{\partial L}{\partial a_k^3}\frac{\partial a_k^3}{\partial z_k^3}\frac{\partial z_k^3}{\partial w_{ki}^3}$

The update should be performed after the network has seen all the training samples in the dataset in order to make the weight update contemplate every aspect of the samples. However, when the training set is populated with millions of samples, each update will take a lot of time as all the samples need to be processed. In order to overcome this limitation, the training is more commonly done using the Stochastic Gradient Descent (SGD) where the iteration is done using just subsets of the training set (i.e., batch). Based on this, the forward and backward pass of the batch through the network is denominated as an iteration, and processing the entire dataset using batches is called *epoch*.

The magnitude of the update is defined by the value of the *learning rate* parameter. A high *learning rate* will make the value of weights in the network take big steps towards the function minima, making the values of the weights abruptly oscillate and causing sometimes the network not to ever reach the minima. Instead, a small *learning rate* will make the weights updates to be very small making the network very slowly approach the loss function minima, and thus, extending the training time. This makes the value of the *learning rate* an important aspect of the network's training. Nowadays, there are different algorithms available for the weight update such as Adam [54], Adagrad [37], or RMSprop [49] that propose different ways of updating the weights and also consider the adaptation of *learning rate* value, making it bigger or smaller as the network gets closer or farther from the minima.

For example, Adagrad adapts the learning-rate for each parameter, making it larger for parameters that do not update frequently, and smaller for those parameters that frequently update. In order to adapt the update for each parameter, the algorithm keeps the square-sum of the gradients throughout the training in a big vector equal size of the number of parameters. When updating the parameters, the learning rate is adjusted for each parameter by dividing it by the value that corresponds to each parameter in the vector. As the square-sum of the gradients will always be a positive number, the values in this vector will grow up each training step making the update shrink until it will eventually turn very small and make the network stop from training. This weakness of storing the gradients is substituted in other algorithms like Adam and RMSprop, which instead of using all past gradients, they define a time window and store the decaying average of past squared gradients. In this way, the update is still performed for each parameter individually, but there is no such risk of diminishing learning rates in later iterations. For a more detailed explanation of the different algorithms available in the literature please refer to the review in [82].

2.2 Metrics

The evaluation of algorithm performance is a key step in the field of Machine Learning. Different algorithms can be used to solve the same problem (e.g., image classification), however, it is important to determine which is the best performing method. For this purpose, there are standardized metrics that describe different aspects of performance which can be then used for an objective comparison. Noted that the metrics that are going to be exposed assume that the task is a supervised learning scenario.

Before starting with the metrics, it is necessary to explain that in a classification problem, being a multiclass or a binary classification, there are four possible ways of evaluating the model output: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). In order to assign one of these labels, it is necessary to first specify which is the positive and negative class in the classification. For instance, in an industrial inspection case, defective parts are usually considered the positive class while a non-defective part will be considered the negative class. Therefore, if the model classifies a part as defective, and it turns out to be defective, the output is defined as True Positive. On the other hand, if the sample belongs to a non-defective part, and is classified as such, this output is considered True Negative. In the case of the sample being misclassified, the defective part classified as non-defective would be considered a False Negative, and the non-defective part classified as defective a False Positive. If the problem involves multiple classes, the same procedure is followed but evaluating one class at a time, where the class to be evaluated will be the positive class, and the remaining classes will act as the unique negative class.

Once the classes are established, the algorithms are executed on the test sets and the results are accumulated in the described four groups. Then, different metrics are employed where the accumulated results are combined in different ways to reflect several aspects of the performance. The metrics that are more widely used are exhibited in Equations 2.3, 2.4, 2.5, and 2.6.

$$Precision : \frac{TP}{TP + FP} \tag{2.3}$$

$$Recall : \frac{TP}{TP + FN} \tag{2.4}$$

$$Specificity : \frac{TN}{TN + FP} \tag{2.5}$$

$$Accuracy : \frac{TP + TN}{TP + FN + TN + FP} \tag{2.6}$$

Precision describes the percentage of success with respect to the positive class. In other words, from all the samples that have been classified as belonging to the positive class, how many were correctly classified. Recall represents the percentage of detection with respect to the positive case, that is, from all the defective samples during evaluation, how many was the model able to detect. And Specificity indicates the same as Recall but with respect to the negative class. Also, there is another metric called Accuracy that

measures the overall performance by taking into account how many classification results were successful regardless of the class.

As mentioned above, these metrics are applied in classification problems where the classification unit is the whole image. Nonetheless, the algorithms might not give a binary classification but rather a probability that the image (or pixel) belongs to a specific class. In order to apply the metrics the results need to be binarized by a threshold. Based on the value of the threshold an output can be considered a False Positive or True Positive. To choose the best threshold, the Receiver Operating Characteristic (ROC) curve is often employed. This curve shows the relationship between the Recall and False Positive ratio (or 1 - Specificity) at different threshold values. Figure 2.5 shows an example of what this curve looks like. The closer the curve is to the upper left corner will indicate that the more accurate the algorithm has been at the classification. When the curve consists in a diagonal line in the middle of the graph (illustrated by the dashed lines in the image), it would indicate that the algorithm is not better than a random classifier. And if the curve falls below the dashed line, it would be the signal that the algorithm has a terrible job. Based on the curve, the best threshold value can be found which will give the highest recall and the lowest False Positive ratio. Along with the curve, the Area Under the Curve (AUC) value, which is independent of the chosen threshold, is usually provided.

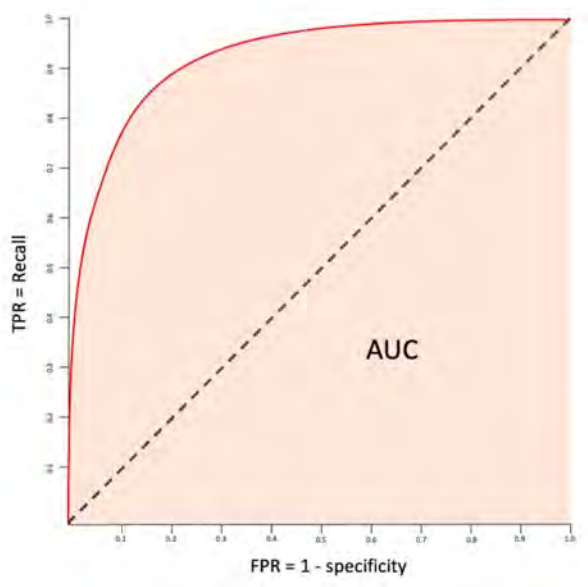


Figure 2.5: Example of a ROC curve

This thesis has focused on locating defects in the cells aiming to provide more interpretable information to evaluate whether cells are defective or non-defective. In image segmentation problems, additional metrics are employed that evaluate the precision of that segmentation. These metrics usually measure the accuracy of the classification taking every pixel in the prediction as an independent classification. Then, the predictions are averaged by the class present in the test image. Also, the pixels with the same predicted class are grouped together forming interconnected areas or blobs. Then, these areas are compared against the ground truth blobs (i.e., defects in the label) in order to evaluate if the defects were detected or not.

In our case, the objective was not to seek a perfect segmentation like in other domains such as the medical domain where an accurate location and shape of a tumor can be key to later design one type of treatment or another. In our scenario, it was sought to provide the users of the methods with additional information rather than the ultimate segmentation. Taking this into account, it was decided to just evaluate the segmentation accuracy of the results at a qualitative level, and use these segmentation results to evaluate the whole prediction as defective or non-defective. Such that, the problem turns into an image classification problem where the described metrics can be used. For example, Figure 2.6 illustrates how a segmentation evaluation can be used to get an overall evaluation of the prediction. Does the prediction contain a certain amount of defective predicted pixels? yes, therefore this cell can be considered defective. Instead, if the prediction is clear or does not contain a sufficient number of defective pixels, the overall evaluation will be that the cell is non-defective.

2.3 Solar panel production

The production of the PV panels is composed of a sequence of different steps that transform the initial solar module into the final solar panel. A finished panel shown in Figure 2.7 is made of the following materials:

1. **External frame:** This frame is the external structure of the panel. It is usually made of aluminum and serves as a seal and protection to the panel.

2. **Crystal:** The crystal is a layer that provides extra protection to the panel against external threats.

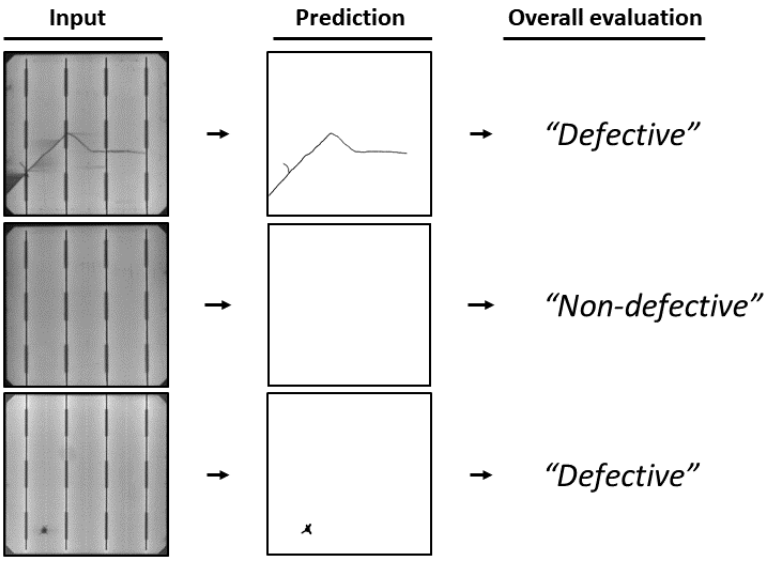


Figure 2.6: Overall evaluation using segmentation results.

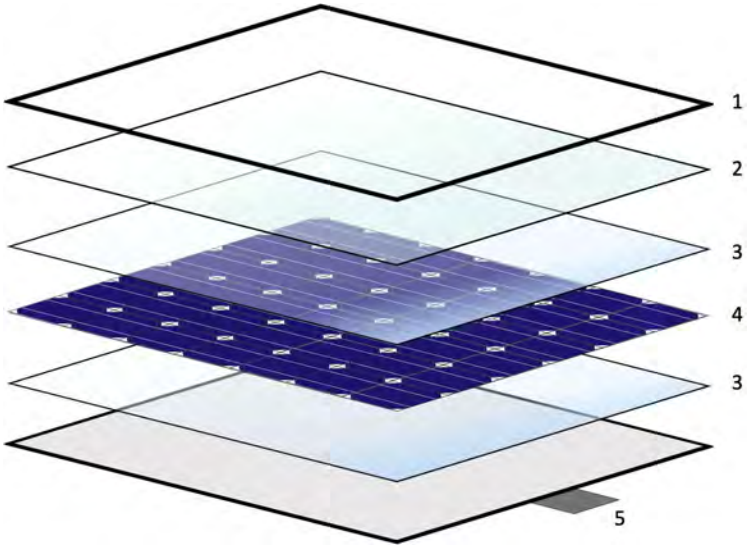


Figure 2.7: Standard solar panel composition illustration.

3. **Encapsulant:** The encapsulant consists of three different parts that ensure a long useful life of the module. The first layer is a solid crystal that protects the panel from threats such as radiation, shocks, or unfavorable weather conditions. After the first layer, two encapsulate layers, one on top and another behind the panel, seal the connections along the cells isolating the panel. The isolation prevents the panel from short circuits. Finally, a back layer seals the encapsulation adding further insulation and protection to the panel.

4. **Cell matrix:** The cell matrix consists of solar cells that are interconnected to each other with ribbons and buses. The ribbon or tab is a wire made of tinned copper that is used to join (i.e., solder) cells such that they compose a closed circuit. The buses or busbars instead are vertical lines drawn in the cells that indicate the position where the wires should be placed and soldered.

5. **Junction box:** The main function of this component is to group together the different connections in the panel as well as protect the entire system from nonuniform radiation which can affect the panel performance.

The cells that compose the matrix are the most important component in the panel as they are the ones that will turn solar energy into electricity. Nowadays there are different types of cells available in the market, however, the most distributed ones are the first generation cells that are made of silicon. These kinds of cells are mainly fabricated following two types of compositions: monocrystalline and polycrystalline. In the case of polycrystalline cells, the cells are composed of small silicon crystals of different sizes that are melted together. In the case of monocrystalline cells instead, the cells are made from a single silicon crystal. This difference makes the monocrystalline cells present higher levels of electricity generation efficiency than the polycrystalline cells but it also implies a higher final cost and a more complex production process. This is one of the reasons why some manufacturers choose polycrystalline cells, that even though they are less efficient at electricity production they allow them to save money. Within each type of cell, the cells are also classified based on the number of buses and the size. Most of the produced cells have a size of 156.75 x 156.75 mm and several buses ranging from 4 to 6. However, nowadays the number of buses has been continuously increasing, reaching 12 and 16 buses per cell.

The mentioned components are commonly assembled in the following steps:

1. **Glass and encapsulant loading:** the manufacturing process starts by loading the front glass and the encapsulant on the production line. Once loaded, they wait in line while the structures that will compose the cell matrix are created in parallel. The loading is usually done by a machine due to the weight and dimensions of the material, however, it is common to have an operator checking the process.

2. **Tabber&Stringer + Layup:** While the glass and encapsulant are being loaded, in this machine the cells are put in lines with the *ribbons* forming what is called a cell array or *string*. Then, the *strings* are grouped into matrices. The number of cells per *string* and number of *strings* per matrix may vary depending on the

final design of the panel. Commonly, the matrices are composed of 6 *strings* with 10 cells per each, which results in a 60 cell panel. After generating the structures, a robotic arm takes and places them on top of the encapsulant. All this is done with the help of machine vision cameras and sensors to ensure that everything is precisely in the right place.

3. **Interconnection:** When the 6 strings are properly positioned in the encapsulant, the module enters into the interconnection machine where the strings are soldered together to form a single electrical circuit.

4. **Lamination:** After connecting all the cells together, the panel goes through a lamination process consisting of three distinct phases. This step ensures the final quality of the module. First, a small pressure and vacuum is applied to seal and fill out possible holes between the components. Secondly, high temperature and pressure cause the components of the different layers to adhere to each other and the air inside to be flushed out. Finally, a cooling phase completes the sealing of the panel.

5. **Edge trimming:** During the lamination process, excess material from the layers may be left sticking out of the panel. This step serves to remove that material such that the panel remains clean for the next step.

6. **Framing:** Finally, an aluminum frame and junction box are placed on the panel to provide rigidity and ease of handling during installation.

Throughout the process, several sources of failures can affect the panel such as components exposed to shocks, excessive pressure when transporting the panel, or mistakes made by the machine when performing any of the previously described assembly steps. These events lead to damaged or defectively assembled components, being the cells the ones prone to present most of the failures due to their fragile structure. To ensure that damaged modules do not end up in the final panel, several quality inspections are carried out during the process. Commonly, two inspections are performed, one before the lamination and another after the lamination. If a defective cell is detected in the former, it is usually manually replaced or repaired, whereas if the detection is done in the latter, the replacement is not that easy to perform. In this case, the panel is usually left with the defective cells and sold at a cheaper price. The detection of the defects is difficult to perform with the naked eye, because of their size (some are usually smaller than 1 millimeter), or because other components in the cell can be mistaken as a defect.

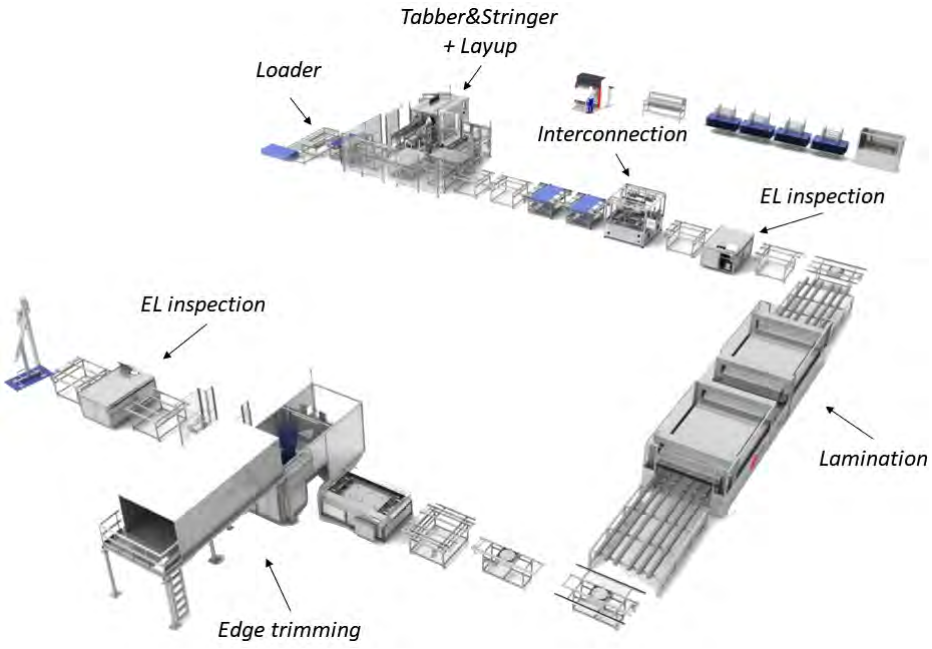


Figure 2.8: Schema of a solar panel production line. The machines that constitute the line depends on the design of the panel to be built and the level of automation to be achieved. Some of the steps require a machine while others can be done manually.

The most common defect that are found are the following:

- **Cracks:** This defect consists of black lines like structures that appear perpendicular and diagonally to the buses. They are usually caused by an excessive pressure that has been applied to cells during transportation or soldering.
- **Microcracks:** Microcracks are a type of crack that are characterized by their size which is usually less than 1cm.
- **Finger interruptions** The fingers are structures that appear perpendicular to the buses and allow electricity to flow through the cell. Due to shocks or failures during soldering, some areas between the buses and these *fingers* can end up disconnected from the main circuit making the electricity flow get interrupted at that point.
- **Bad soldering:** When soldering the cells, the soldering may not be well done. This can result in areas adjacent to the buses (usually external ones) not conducting electricity properly.
- **Breaks:** This defect is just a complete fracture of the cell that could be caused by a more severe physical shock.

- **Black spots**: In the same way that an excessive pressure can cause cracks and microcracks, or in a more extreme situation, a break, the same kind of events can also cause very localized points to get disconnected from the cell. Instead of longitudinal line appearance, black spots are more like isolated dots that interrupt the electricity flow.

A defect that covers 8% of the total module may not have a major impact on performance if this module is isolated; this same area can have a significant impact when the cells are connected together, which is the usual structure [59]. Because of this, a strict criterion is usually established during the quality inspection aiming to detect as many defective cells as possible. For example, a cell can be rejected if it contains an inactive area greater than 10% of the cell that has been caused by a crack, microcrack, or a finger interruption. Or also, require to be put aside and soldered again if less than 80% of the busbar area is not correctly soldered before the lamination. These measures ensure that defective cells will be removed and finished with a high quality final panel.

These defects are difficult to be detected by the naked eye, therefore, various techniques are often used to help detect them. Numerous inspection methods are currently available such as Electroluminescence, Thermography (IR), Photoluminescence, acoustic vibration based detection, or electrical modeling (i.e., measuring electrical generation capacity). These techniques allow practitioners to obtain enhanced images or measurements where the defects themselves or their effects are highlighted. The techniques are chosen based on how intrusive they are or by the environmental conditions where they need to be employed. For example, both Thermography and Electroluminescence output images that highlight the location of the defects but they are used in different contexts.

In the case of Thermography, it takes advantage of the temperature difference that a well working cell and a defective cell will present. The cells are connected in series in the panels, therefore if one cell is defective the electrical current cannot flow properly through the circuit affecting the overall power generation capabilities of the panel. To avoid this, bypass diodes are often installed allowing the current to avoid passing through these defective cells. However, sometimes this bypass does not work properly, causing the current to concentrate on the defective cells and increasing their temperature. By capturing IR images, these hot-spots can be easily located in the panels.

Instead, the Electroluminescence technique takes advantage of the physical phenomenon of Electroluminescence that the cells show when are put under an electrical current. Solar panels are designed to capture the energy coming from the sun and transform it into

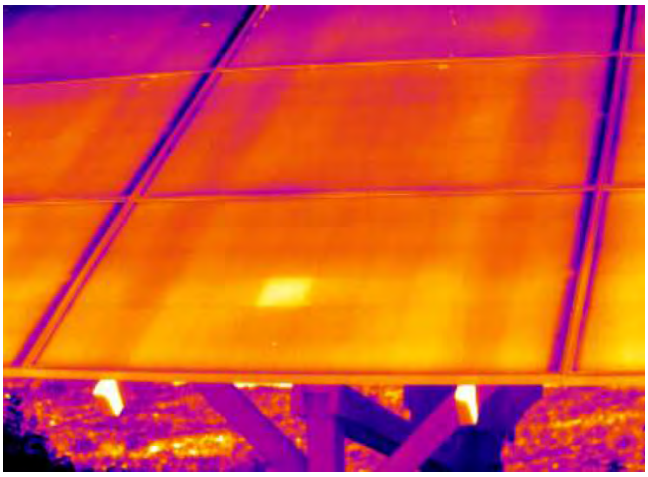
electricity. In this technique, the energy flows the other way around. The solar panels are connected to an electrical current making the solar cells emit light. This light is captured in high resolution images showing the areas that are capable of conducting electricity and those that are not. Bearing in mind that in the cells only those areas made of non-conductive material should be the ones that do not allow electricity to flow, it would be possible to detect those defective areas that will present the same non-conductive behavior. Figure 2.9 presents two examples of images captured using the IR and using EL.

However, Thermography is more commonly used outdoors for inspecting already installed panels, while Electrolumiscence is more commonly used for indoor industrial inspection. Thermography imaging allows us to rapidly visualize those cells that emit anomalous levels of temperature. By using drones or Thermographic cameras coupled to cars, wide farms of PV panels are inspected [36, 42]. Nonetheless, IR images will only show which of the cells in the panel exhibit high temperature values, but do not show the specific location of the defect within the cell. On the other hand, EL images are more precise regarding the location of the defects, but the process of capturing the images must be taken in total darkness. This limits its application outdoors but makes it applicable in industrial inspection where there is greater control over environmental conditions [78, 99].

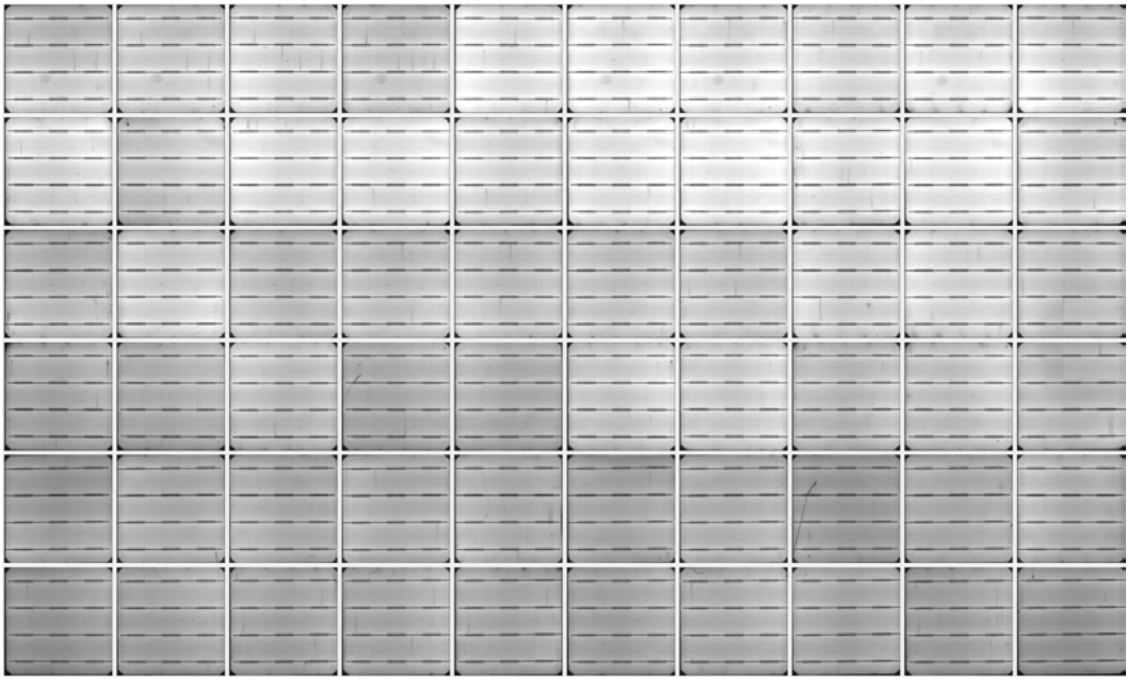
This thesis has focused on cell inspection using the Electroluminescence technique as one of the most widespread techniques for quality inspection during production and also because there has been access to EL images extracted from a real environment. This has allowed us to validate the performance of the techniques that have been explored in real conditions.

2.4 Datasets

For the experiments in the thesis, three different industrial datasets were provided by Mondragon Assembly S.Coop which contained Electroluminescence images of monocrystalline and polycrystalline solar cells, all extracted at the quality inspection stage from real production lines. The following sections will focus on describing the characteristics of the data in each dataset.



(a) Example of a Thermograhy image of a panel, source: [103].



(b) Example of an Electroluminescence image of a solar panel.

Figure 2.9: Examples of different imaging techniques used for solar cell inspection

2.4.1 Electroluminescence

Electroluminescence is a physical phenomenon that happens to certain materials which emit light when they are under forwarding bias conditions (i.e. electrical current flow through them). The light that they emit is near infrared light which is in the wavelength

range of 780-2,500nm. But more specifically, the wavelength of EL is usually in the range of 1,000-1,200nm with a peak around 1,150nm [41] Figure 2.10.

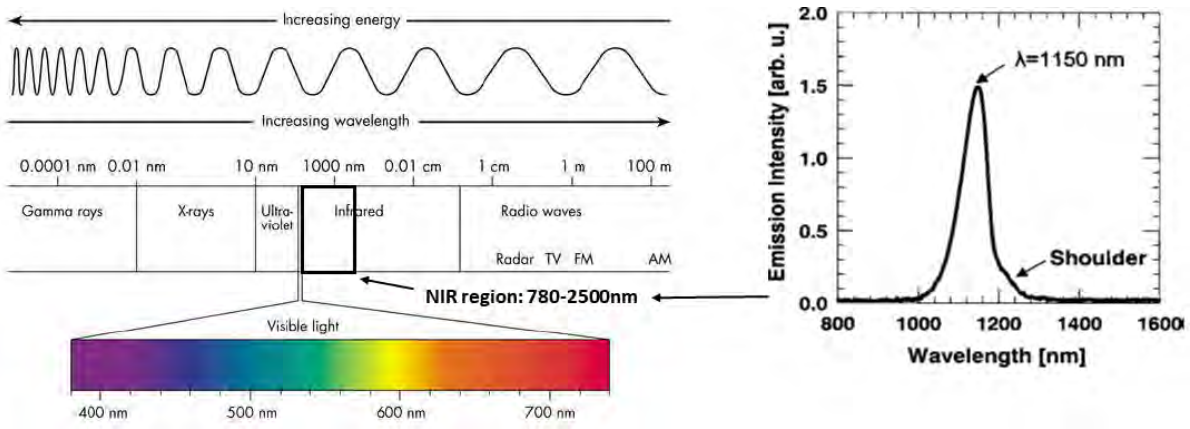


Figure 2.10: Full light spectrum diagram and typical emission spectrum of EL. Image source: [6] and [41]

These numbers consist in values that well functioning cells present. However, there might be different causes like defects that can alter these numbers and present deviations in how the light is usually emitted. These variations can be captured using a cooled Si-CCD camera and turned into high-resolution grayscale images using a set-up like the one illustrated in Figure 2.11.

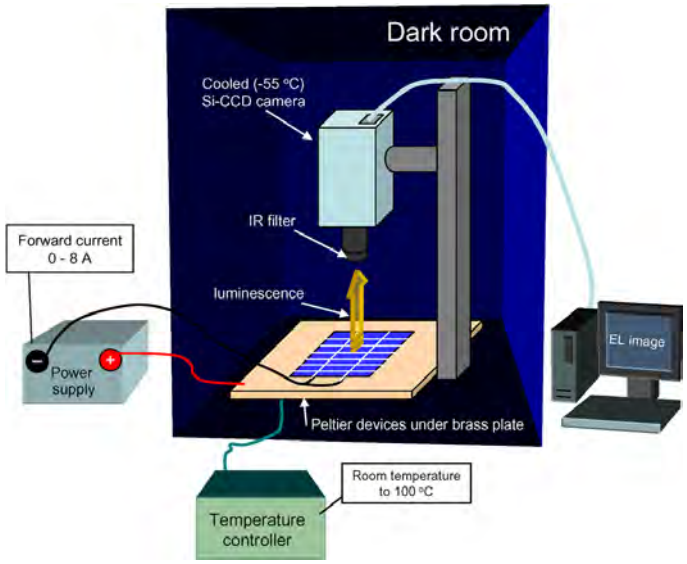


Figure 2.11: Simple set-up to capture EL images from solar modules. Image source: [40]

These EL images reveal spatially resolved information about the electronic material properties of solar cells as well as defects or breakages in them [40]. Usually, during

the industrial inspection, the whole panel is processed using an industrialized version of the set-up in Figure 2.11, and EL images from all the cells are captured. Then, the images are put all together to inspect them to find defective modules. As mentioned, EL images are in a grayscale where areas that present electrical conductivity appear with a light color while areas that do not conduct that will appear with a darker color. This difference is used to spot areas that in advance are known to have a light color in the images but appear dark. Figure.2.12 illustrates some EL images with different defects that might appear during production. The inspection process is still done by humans to some extent. However, efforts are being done to automate the process mentioned before and described in the next chapter regarding the techniques proposed for such purpose.

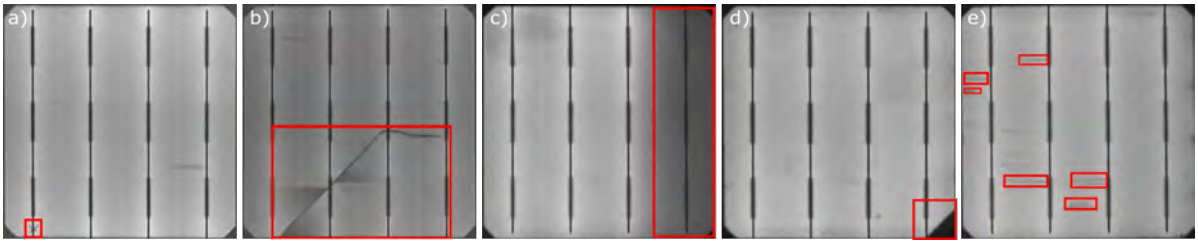


Figure 2.12: EL images of different defective cells. The defects in the figure are a) microcrack, b) crack, c) bad soldering, d) break, and e) finger interruptions.

2.4.2 Polycrystalline cells

The first dataset is composed of 542 EL images of **polycrystalline cells** of 15x15cm that have an average resolution of 943x923 pixels. As stated before, polycrystalline cells are made by melting together many silicon fragments to form the wafers that compose the silicon panel. This procedure makes the cells have a heterogeneous background as can be seen in Figure 2.13.

The dataset contains defective and defect-free cells with 3, 4, and 5 buses, distributed as illustrated in Table 2.1. Defect-free samples are samples that were in proper conditions, and thus, considered to be part of the final solar panel. Whereas the defective samples contained one or several defects that were severe enough to be discarded from the final panel. The classification of whether the individual cell should be defective or defect-free was done by the company based on their quality criteria.

Regarding the type of defects, the cells in the dataset contain mainly cracks. Some of the cracks are severe and easily visible, while there are other smaller cracks (i.e.,

	Total
Defect-free	397
3 Buses	7
4 Buses	189
5 Buses	201
Defective	145
3 Buses	38
4 Buses	100
5 Buses	7

Table 2.1: Sample distribution in the polycrystalline cell dataset.

microcracks) that are harder to be detected. Figure 2.13 shows some of these defective samples side-by-side with defect-free samples.

The provided dataset also included a ground truth image with a pixel level {0,1} annotation for each individual cell image. As can be appreciated, if the crack is not severe, in certain defective samples cracks can be confused by grains and vice versa. Thus, in some cases, it might not be so trivial to classify the cell either as defective or defect-free. Even though this can impact the performance metrics of the models, this has not been considered as part of the scope of the thesis, thus the labels provided by the company have been considered the ultimate ground truth for both models' training and evaluation.

Based on the labels, several statistics have been calculated to show the severity of the defects in the cells. The computed statistics have been the average number of defective pixels, the standard deviation of defective pixels, and the ratio between defective and background pixels for the samples with severe cracks and microcracks. All metrics are shown in Table 2.2.

	Crack	Microcrack	Crack (severe)
avg def. pix.	4,782	726	16,452
std def. pix.	4,150	385	11,456
ratio def./def-free	0.51%	0.07%	1.7%

Table 2.2: Average defective pixels per type of defect in polycrystalline cells. Taking into account the size of the cells, a microcrack of about 50 pixels length can be around 0.75cm if its diameter is 1 pixel. Usually the diameter is about 3-5 pixels.

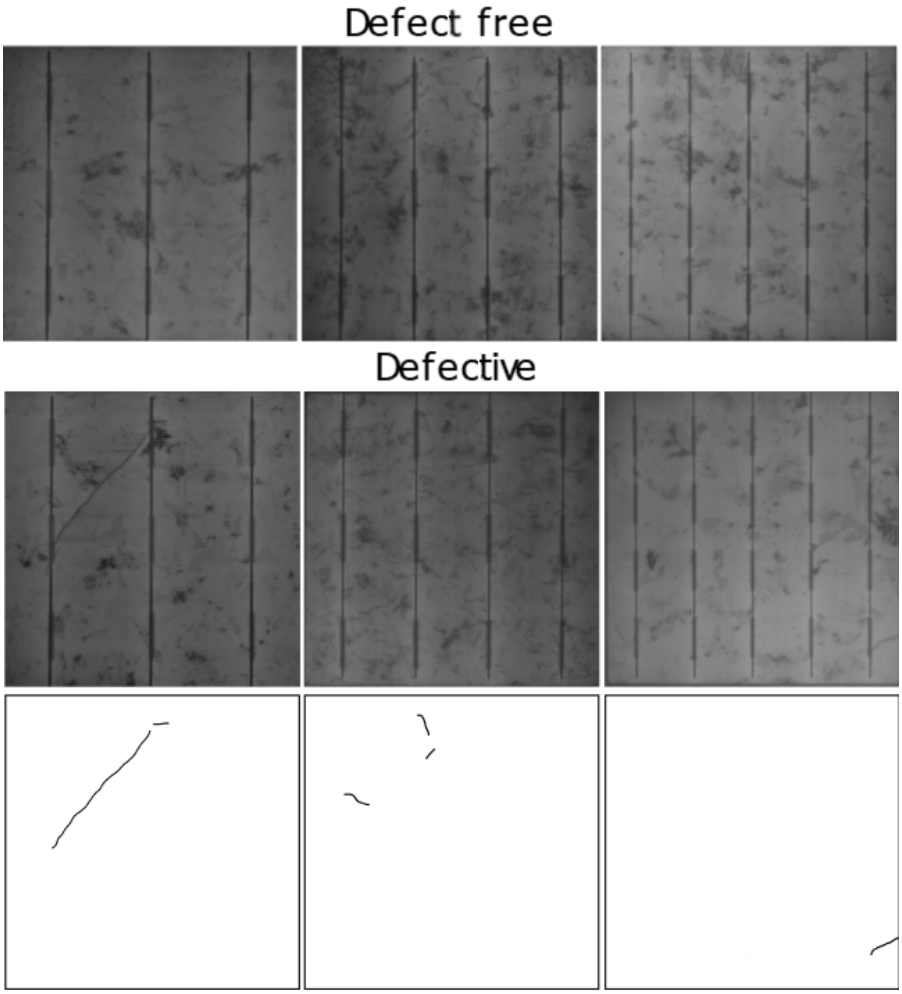


Figure 2.13: Cell samples from the polycrystalline dataset and their corresponding ground truth.

As can be seen in Table 2.2, there is a great unbalance between defective pixels and background pixels in the images, which can considerably affect the training and final performance of the models. This is more notorious in the case of samples with microcracks, where the microcracks appear in groups of two or more. Thus, the average defective pixels in the table for these samples should be divided by two or more in order to have the real ratio between defective and non-defective pixels in the images. The ratios in Table 2.2 have been calculated to try to convey to the reader the difficulty of the segmentation on these data samples. Mainly to keep it in mind when analyzing the results from the experiments.

2.4.3 Monocrystalline cells

The second dataset consists of 1886 EL images of **monocrystalline cells** with an average resolution of 840x840 pixels. As mentioned in Section 3.3, the monocrystalline cells are generated from a single silicon wafer which leads to cells with a more homogeneous background than the polycrystalline cells as can be appreciated in Figure 2.14. This dataset also contains samples evaluated as defective and evaluated as non-defect. In this case, the cells are more uniformly distributed as they only contain 4 buses. Within defective samples, there are 5 different types of defects: cracks, microcracks, finger interruptions, black spots, and bad soldering. Table 2.3 shows the sample distribution in this dataset, where it is appreciable that there is a great unbalance of samples for each type of defect.

	Total
Defect-free	**1,498**
Defective	**388**
Crack	18
Microcrack	240
Finger interruptions	117
Breaks	2
Black spots	7
Bad Soldering	4

Table 2.3: Sample distribution in the monocrystalline cell dataset.

The dataset is mainly composed of samples with cracks, microcracks, and finger interruptions which are illustrated in Figure 2.14 along with a couple of defect-free samples.

With regard to the black spots, breaks, and bad soldering defect classes, the dataset only contains few samples for each of them. Some of the available samples are shown in Figure 2.15.

These latter defects do not appear as frequently as crack, microcracks, and finger interruptions. Moreover, the company labeled them as hard defects giving priority to the former ones. Because of the lack of samples as well as having a lower priority for the company, the training, and evaluation in the experiments have been focused on the samples with cracks, microcracks, and finger interruptions while leaving the samples with breaks, black spots, and bad soldering for qualitative evaluation.

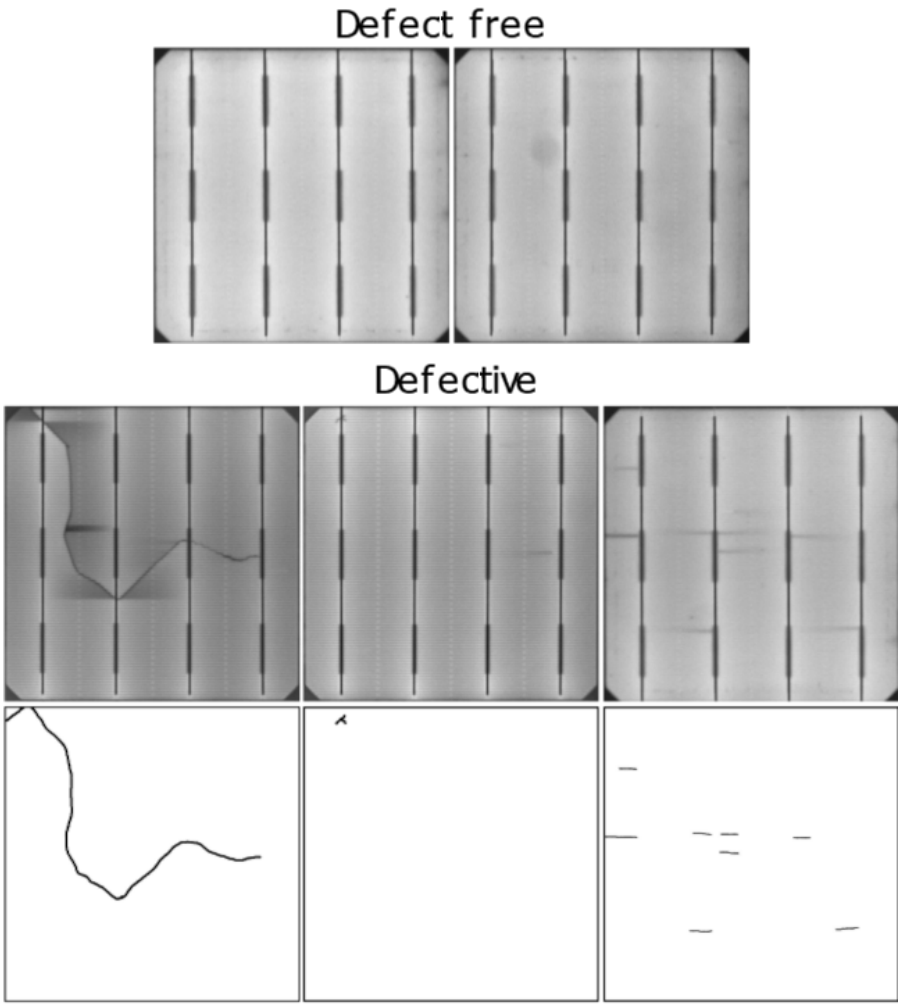


Figure 2.14: Samples with cracks, microcracks and finger interruptions from the monocrystalline dataset and their corresponding ground truth.

For every defective cell, a pixel level annotation indicating the defects has been created. The following Table 2.4 shows some statistics about the defects present in the cells.

	Crack	Microcrack	Finger Interruption
avg def. pix.	3,546	398	887
std def. pix.	2,410	446	675
ratio def./def-free	0.50%	0.056%	0.17%

Table 2.4: Average defective pixels per type of defect in polycrystalline cells.

As in the previous dataset, here also the defective pixels represent a tiny percentage of the whole cell. The numbers show that the samples with finger interruptions present more

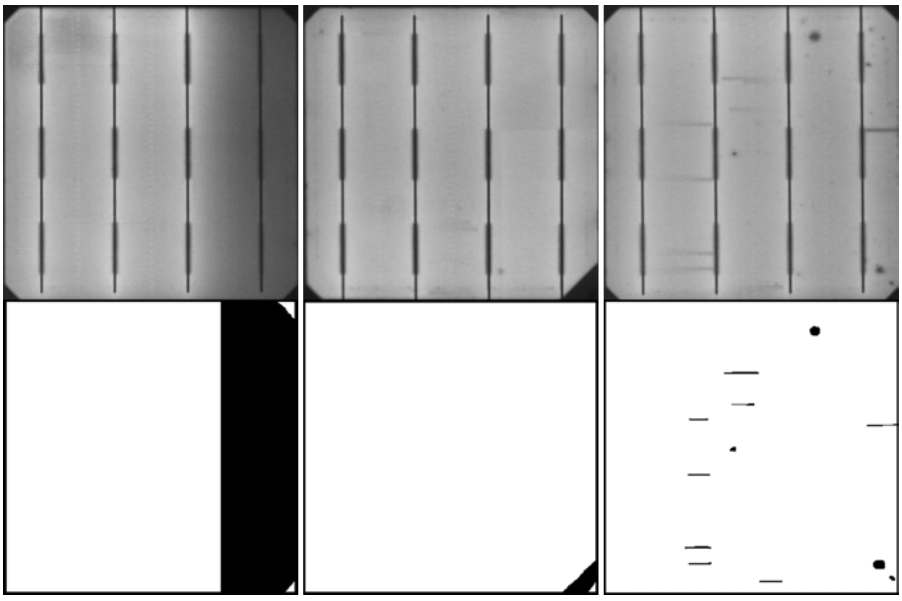


Figure 2.15: Samples with bad soldering, black spots, and breaks. and their corresponding ground truth.

defective pixels than the samples with microcracks. It should be noted that microcracks commonly appear isolated while finger interruptions usually appear in groups of three or more, thus, they contain more defective pixels.

2.4.4 Sequence of 700 panels

At the beginning of the section, it was mentioned that we were provided with three datasets for the development of the thesis. Of these three datasets, the two described in the previous subsections are the main datasets used in the experimental phase. However, to validate the proposed methodology an additional dataset was employed in the very last experiment in the thesis.

This third dataset is composed of images taken from a set of solar panels of monocrystalline modules. The set consisted of a set of sequentially produced 700 panels, each composed of 60 monocrystalline cells, which in total counted for 42,000 monocrystalline cells. In contrast to the other datasets, the cells in this dataset were not manually labeled. However, the cells were reviewed and it was seen that about 70% of the cells were completely defect-free and the rest of the cells have similar defects to the ones shown in subsection 2.4.3. Regarding the defect class distribution, the vast majority of them were finger interruptions instances, which corresponded to about 95 percent of the defects.

Then, there were also cracks and microcracks but in more moderate quantities, 77 and 405 respectively.

2.5 Hardware and Software specifications

All the experiments in the thesis have been performed in a server located in the university which is equipped with 40 CPUs and 3 GPUs (2 GeForce RTX 2080 Ti and 1 TITAN Xp), nevertheless, the maximum number of GPUs required during the training of the models was 2. For more details about the hardware requirements in each experiment check the section in chapters 4, 5, and 6.

Regarding the software, the server has Linux as the operative system, and on top of that, several Docker containers for each application or user. Docker is a platform that allows launching different denominated Containers, which consist in software packages that act as standalone environments providing all required dependencies for the application to run but isolating them from each other. By isolating environments, the applications will access the same hardware in the server but without the need to deal with dependencies conflicts. In this thesis, three different docker containers have been used which were created from three different Docker images as illustrated in the Figure 2.16 schematic.

The first one has been created using the image available at Nvidia's docker repository with the tag *nvidia/cuda:10.0-cudnn7-devel-ubuntu16.04*. The image comes with CUDA software toolkit version 10 and CuDNN version 7. Moreover, all the neural networks in this contained were developed with Python 3.6 using Tensorflow and Keras at their version 1.14 and 2.3.1, respectively. Also, well-known libraries such as Numpy, Pandas, and Matplotlib were also employed mainly to work with the images and analyze the results from the experiments.

In the case of the second container, the image used can be found in the same Nvidia repository with the tag *nvidia/cuda:8.0-cudnn7-runtime-ubuntu16.04*. This second container was created to work with CUDA 8 as the software downloaded from the repository github:f-AnoGAN was done using this CUDA version and python 2.7. As it was wanted to use the original version and experiment on top of that, it was decided to keep the same version of CUDA and python. All the code could be migrated such that it could be possible to work with the previous container with CUDA 10 and python 3.6. However, there was a chance of introducing some possible bugs that could impact the results so it was decided not to try any migration. The neural network from the

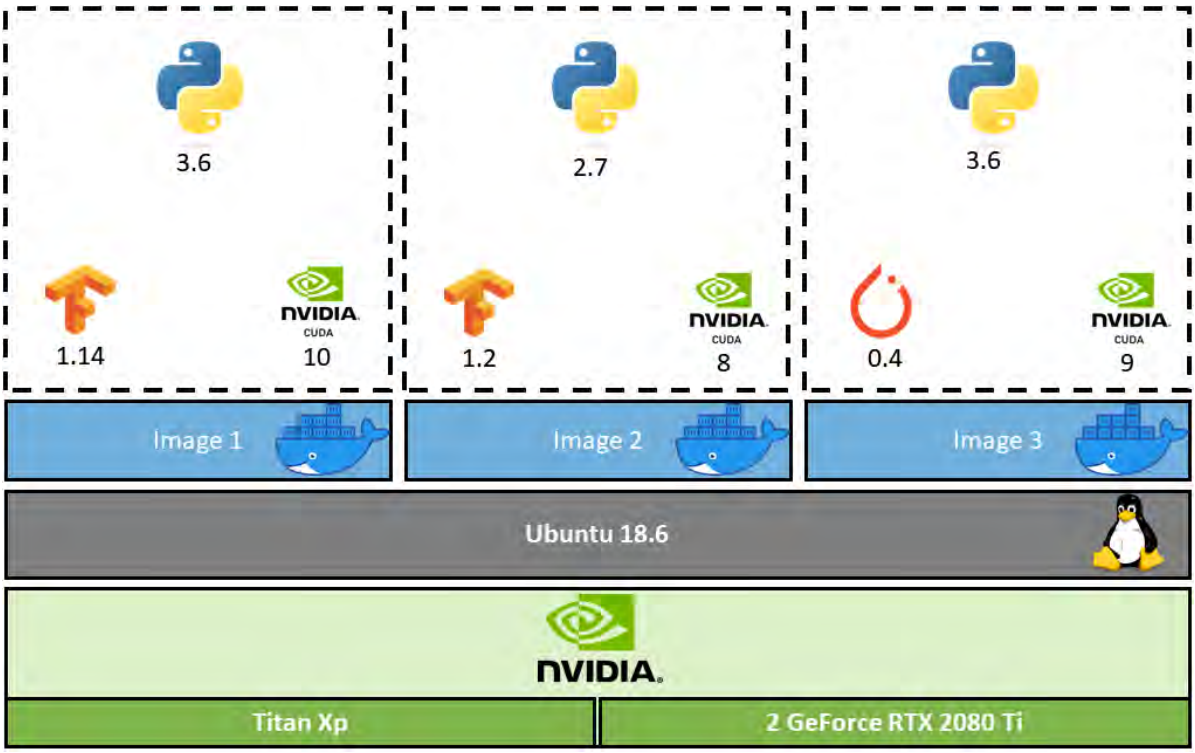


Figure 2.16: Schema of the main Hardware and Software used in the thesis.

repository was programmed with Tensorflow 1.2, so it was also kept at that version. With respect to the other libraries, Numpy, Pandas, and Matplotlib were also the main libraries that were used to work with the images and data.

Finally, a third container was also created using the script available at AMPproxies repository. The Docker is created using the base image with tag *nvidia/cuda:9.0-cudnn7-devel-ubuntu16.04*, also from the same NVIDIA docker repository mentioned above. The code that was used in the experiments was from the same repository, so to avoid any source of problems it was decided to work with that image instead of any of the previously mentioned images. Regarding the libraries, the experiments with the neural networks were done using Pytorch as all the code was done with it, and as in the previous two containers, Numpy, Pandas, and Matplotlib were used to work with the images and data.

Chapter 3

Literature review

In this section, a summary of the state of the art of different image processing techniques used so far for the detection of defects in solar cells will be provided. In this thesis DL methods have been mainly applied, however, it has been found necessary to outline the different techniques that have been used in some works in the field and highlight their characteristics in order to reason why the choice of DL methods for this thesis.

The techniques can be divided into three groups based on the level of human intervention required for their application: 1) traditional image processing methods, all procedures such as feature extraction design or decision making rules are manually defined, 2) traditional image processing methods in conjunction with traditional Machine Learning methods, where the decision making step from the previous procedures is replaced with Machine Learning techniques, and 3) Deep Learning based methods, where all feature extraction and decision-making is automatically done using end-to-end trainable neural networks.

3.1 Traditional Image processing

The initial works that can be found in the literature regarding defect detection in solar cells are mainly traditional image processing based approaches. Traditional approaches in this context refer to approaches that heavily relied on manually defined feature engineering procedures such as image filtering, morphological operations, or thresholding.

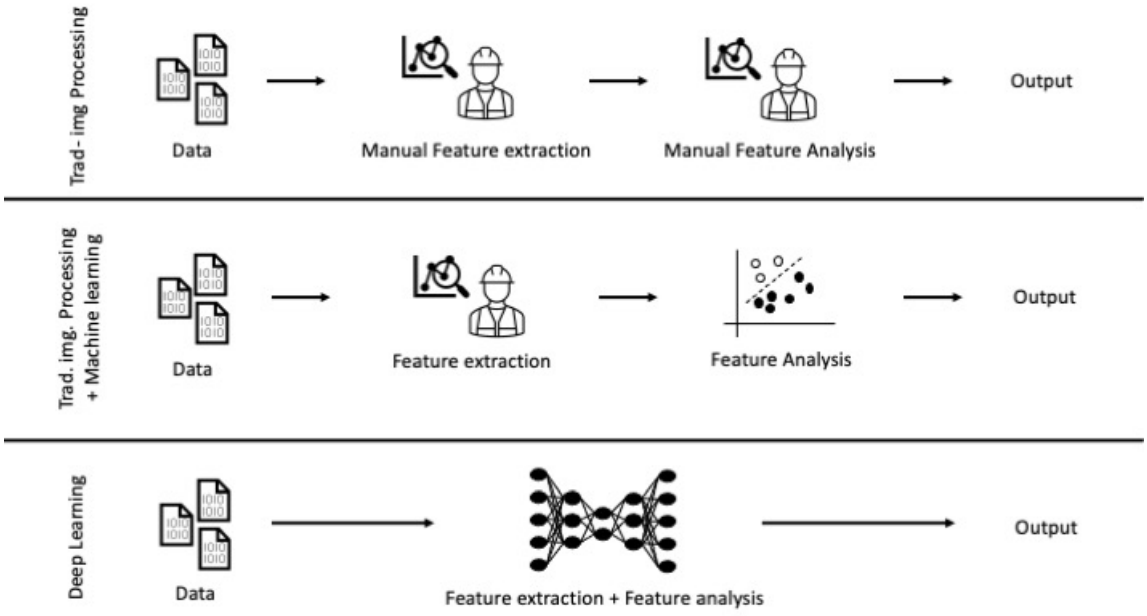
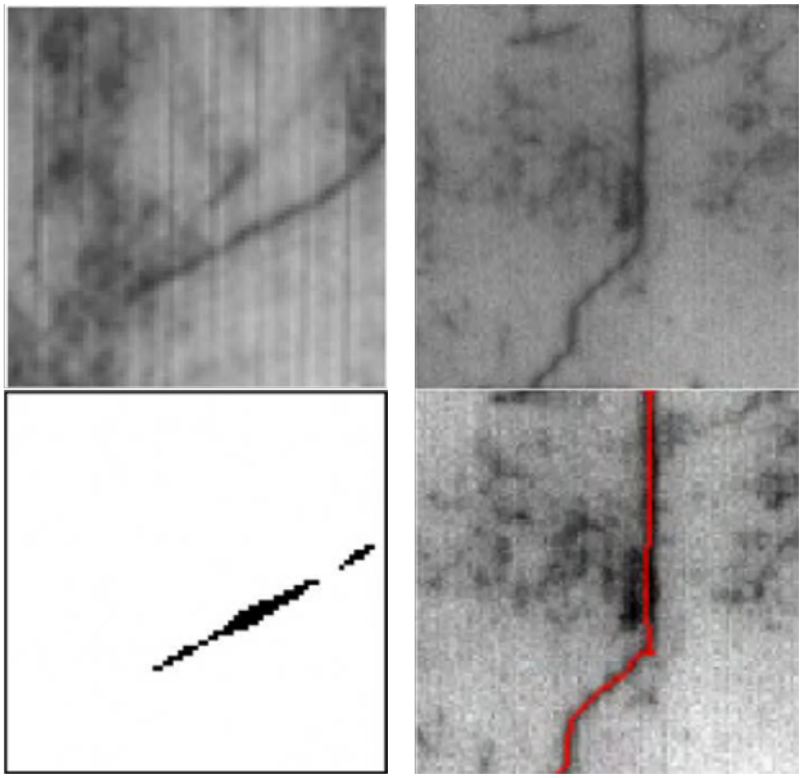


Figure 3.1: General schema of the different approaches found in the literature to develop an inspection system for the quality inspection of solar cells.

The works in this section propose data processing pipelines that are mainly composed of an initial noise reduction operation, followed by a feature enhancement procedure, and finally, the result "cleaning" procedure using thresholding mixed with morphological operations that isolate defects in the cells.

The first step of the pipeline is usually focused on removing the noise (i.e. little structures in the cells) that can interfere with the subsequent steps in the pipeline. What can be considered noise can vary from one scenario to another, going from features that are part of the cell itself like gridlines that appear as horizontal lines in the images, or external objects like dust particles that might have deposited on top of the cells while acquiring the images. As the noise reduction is very case specific, the works found in the literature present several alternatives to deal with it like: Gaussian filtering [120, 94, 110], median filter [112], mean shift algorithm [95], Anisotropic diffusion [55, 97], or Fourier Transform [98, 90]. Even though it might seem that techniques are applied independently from each other, they are usually sequentially employed in order to improve the results.

Once the noise is removed, the next step in the pipeline focuses on enhancing the features left in the images such that, the final step in the pipeline can easily isolate defective features from non-defective ones. As with the noise reduction, the works in the literature show different alternatives for this step: wavelet 2D decomposition [63], histogram



(a) (b)

Figure 3.2: Some result examples from traditional methods based approaches. On top, the input image, and on the bottom the result from the algorithm. (a) Source [98]; (b) Source [25]

equalization [57], mamdani fuzzy logic [19], Parameter optimized Atmospheric Scattering Model (PASM) [23], or Hessian matrix based structure extraction [25].

Finally, the last step after all the preprocessing steps consist in isolating the defective features from the rest. If the images have not shown many peculiarities, this final step can be undertaken with a simple thresholding [88, 56, 112, 98]. If the previous steps have not completely removed all the noise additional basic operations like erosion or dilation are performed [24, 57, 90].

Traditional image processing based approaches are fast and have high precision in detecting defects. Setting aside the cases of [56] and [23], the works reported accuracy numbers above 90% and execution time below one second. Also, the type of cell that is mainly employed in the works is the polycrystalline cells, that as the reader can guess by checking the images in Figure 3.2 makes the problem harder than if the defect detection was to be done in monocrystalline type of cells. In addition, each step that composes the pipelines is simple enough to understand, which is an important aspect to have for

cases where something starts not working as expected as it can help to fix the problem. Moreover, simple approaches do not demand high hardware requirements which is also a positive aspect regarding their deployment.

Nevertheless, these works are ad-hoc solutions that take advantage of case-specific characteristics in order to adjust the parameters for each of the techniques they employ, which can lead to procedures that might lack the flexibility to adapt to current dynamic industrial scenarios. For example, in [97, 94, 55] they assume based on their data that the gray level of the cracks is always much lower than the gray from the crystal grains. Thus, they take advantage of such gradients to isolate cracks from the background and crystal grains using a threshold. However, it is quite common to capture images where the cracks and the crystal grains have very similar gray values due to uneven illumination. Or for example, in [63] they take advantage of the different blurriness of saw-mark edges concerning sharp grain edges in the background, which limits the detection for the cases where the crystal grains present such sharp edges. Or also in [120, 57, 22], they employ case-specific morphological operations tailored to the specific shape of the defects in the cells to enhance them for detection. Also, it should also be mentioned that the works have focused mainly on cracks type of defect. This defect is one of the most common types of defect that can be found during production, however, there are other kinds that also appear and with these solutions could not be detected.

If the production remains unchanged and very stable for a long period of time, may be worth spending time and resources developing the procedure to extract very case specific features that will serve during the inspection. However, in the current dynamic scenarios, as well as the inherent variability that the production is usually subject to, these solutions may have a tough time adapting to such changes.

The traditional methods usually require high domain expertise in order to define the optimum combinations of feature extractors that can retrieve as many defective areas as possible from the cells. This can be considered as another limitation when choosing these kinds of approaches as in some cases, the definition of the pipeline can require more creative-artistic skills subject to the practitioner than just technical expertise.

3.2 Traditional Image processing and Machine Learning algorithms

In the previous section, different examples of manual feature engineering based approaches have been presented. In these works, all the procedure is manually defined, both the feature extraction and then the decision-making based on these features. This way of defining the pipeline can be sometimes cumbersome. In order to alleviate some of the manual labor, some later works incorporated ML methods into the pipeline aiming to automate the decision-making step. In this case, the feature extraction will still be manually performed, but once the features are extracted they will be used as training data to train ML algorithms. After the training is completed, the pipeline will be composed of an initial feature extraction procedures, like the ones seen in the previous section, and a decision-making step composed of a trained ML algorithm that will determine whether the extracted features are defective or not.

The techniques that have been found in the literature regarding this purpose have been classifiers like SVM [3, 32] or AdaBoost [15], clustering methods like spectral clustering algorithm [101], fuzzy C-means [99] or K-means [91], or other feature analysis techniques like Independent Component Analysis (ICA)[95, 116].

The works found for this section have reported overall defect detection accuracy rates above 90% which is similar to the ones in the previous section. Regarding the processing time, some of the works reported times above a second per image which will not meet the industrial inspection speed. For example, [32] reported 12s per 300x300 pixel image or [3] that required 4s per 1,178x1,178 pixel images. However, the rest of the works reported processing time below the 1 second mark per cell, from which it can be concluded that even ML algorithms might introduce more complexity to the pipeline overall but still can meet the industrial processing time limit set. Nevertheless, it should be noted that these algorithms can often find relationships between features that a manual analysis may miss, which sometimes can improve the final detection rates. In this case, practitioners should deal with this trade-off between execution time vs. complexity and accuracy.

Despite reporting high detection rates, the feature extraction that will serve to feed the ML algorithms is still manually defined which does not overcome the lack of adaptability that the works in the previous section present.

3.3 Deep learning based proposals

In more recent works, the traditional algorithms have been replaced by Deep Learning methods. Unlike in the previous two sections, the works in this section employ Artificial Neural Networks (ANN) which automate both main steps in the pipeline, i.e., feature extraction and decision-making.

The ANNs are able to extract meaningful features from the data, as well as approximate non-linear functions, that will serve to perform tasks like image classification. However, in contrast to the previous approaches, this process is done through training where the network learns on its own to select the best combination of filters that will lead to the best performance for the given task. This characteristic will avoid manual work, which is a great advantage from the point of view of the flexibility of the model. Thus, users can adapt the models to variations in the environment in a systematic and simple way, avoiding manual redesign and choosing the best combination of filters and feature extractors for the new circumstances.

After collecting and reviewing the works that apply DL methods for solar cell inspection, it has been seen that the works focused on either image classification or image generation. If compared to the traditional approaches, ANN training is a more data intense procedure. The works in the literature that employed these kinds of methods for image classification employed at least 1800 images for training [107, 92, 11, 26, 71, 32, 1, 64]. This requirement is hard to meet in industrial scenarios where defective samples are usually scarce. Moreover, even having access to a large number of defective samples, will still require an annotation process which is an arduous task to accomplish. This may prevent practitioners from having access to inspection models at an early stage and delay production line set-up. However, if not enough representative defective samples are gathered, the training with few defective samples can lead to overfitting which will increase the risk of experimenting a performance drop during inference. Despite this limitation, Neural Networks avoid the manual engineering process as they automatically learn to perform the classification, thus are more adaptable to new scenarios. Some researchers tried to overcome such limitations by employing generative networks to generate synthetic data that can be used to populate data scarce datasets [27, 96]. Then, both synthetic data and real data are used for image classification.

The proposals in this section were mainly focused on cell classification while some works in the previous section also considered the location of the defect as part of the results.

By the time the thesis started, just some works such as [96, 71, 32] took advantage of the features from the networks as an approximation to the location of the defects. Unlike more traditional methods, neural networks can sometimes act as a black box that receives inputs and outputs whether a cell is defective or not without providing any additional information about the reason behind the results. The pipelines in the previous sections were "simple" enough to debug them in order to find any source of error if a given output is not convincing, reaching higher levels of interpretability about how the model is working. Instead, in neural networks, the feature extraction is designed automatically throughout the training process, and even once the training has been completed it is usually hard to keep track of all operations performed inside the network. Thus, it is complicated to try to debug the model in order to find any sort of explainability about how the network is working. The main task to be addressed during the quality inspection is believed to be the classification of the cells so in the end defective cells can be discarded from production. However, it is thought that providing additional information with the results such as the location of the defects within the cells, might help interpret the results and also evaluate the severity of the defects. Later, as the thesis progressed and the results obtained were published, different works were also published [26, 78, 77, 76, 118, 53] that shared our point of view and also provided defect location to improve the interpretability of the results.

With the only exception of [1] where they designed a light CNN to work on a CPU, in the rest of the works GPUs are employed for training and for inference of the networks. Neural networks require more intensive computation procedures during training than traditional pipelines. That is one of the main reasons why GPUs are commonly used when working with DL methods as they accelerate the training. Nonetheless, nowadays there are cloud services that provide access to multi GPU stations to train the models, and then at inference, the heavy computations can be delegated to a modest GPU incorporated into the inspection infrastructure.

With regard to the accuracy rates and execution time reported, it can be observed that there are a variety of results. Works like [100, 118, 21, 67] reported high accuracy rates (above 90%) and execution times below 1s per image which is comparable to the proposals from previous sections. But there are also other works like [71, 11, 26], where the accuracy rates are lower than the traditional approaches based works. What can be underlined here is that even though works mainly focus on cracks, more types of defects have been taken into account for the classification. Nevertheless, a direct comparison between works is hard to perform as they used different datasets for evaluation.

3.4 Summary

After analyzing the literature, several approaches for developing an image processing based automatic inspection system for the quality inspection of solar cells were found. On the one hand, there are traditional image processing based methods, which are accurate at detecting defects within the images, fast, and are not very hardware demanding. However, they consist in case specific solutions that lack in adaptability, a key characteristic for nowadays rapidly changing scenarios. On the other hand, there are DL based methods, which can also be fast, accurate, robust, and flexible to changes. However, they require a great amount of annotated defective samples for training and have been mainly focused on classification.

Taking into account the pros and cons of the different proposals, it was decided to focus the thesis on exploring DL methods to develop an inspection system for solar cell quality inspection taking into account that: 1) in industrial setups the common scenario is to only have access to few annotated defective data samples (e.g. less than thousands of samples usually required when employing conventional neural networks supervised training), and 2) the techniques should provide a localization of the defects to ease the interpretation of the results. When experimenting with the different methods, it has been always taken into account their applicability in real industrial scenarios with regard to the method's accuracy and execution time.

Chapter 4

Supervised training

Among the different learning approaches in Deep Learning, supervised learning is usually the one that yields the most accurate results. This happens due to the use of labeled samples as training data. When employing labeled samples during the network's training, the networks are being told to specifically search for particular patterns that will serve to discriminate and detect annotated features. For example, in the case of quality inspection, these patterns could refer to defective features in the pieces.

Once the training finishes, the model will search for specified features in test samples. However, supervised learning of DL methods have shown that the accuracy of the results is proportional to the amount of training data employed in the training. Few training samples can increase the risk of overfitting the training data, thus, making the model perform poorly at inference.

Nonetheless, in an industrial context, access to enough defective data samples for training is not always ensured. Usually, in industrial environments, there are not so many defective samples available. Moreover, these samples need to be annotated in order to be applicable as training samples. The scarcity of representative defective samples could be a potential limitation for the applicability of DL methods.

Despite this, in this thesis, there was access to defective samples that were labeled by experts. Taking into account this, it was thought that it was better and more straightforward to check the applicability of supervised learning models on our datasets. Even, not having thousands of data samples, overfitting the data will at least show that

DL methods could behave correctly in our dataset. Once checked the results, it would be possible to try to overcome the lack of defective samples as well as explore approaches to complete the remaining stages of the methodology.

This first chapter will describe two possible supervised learning approaches. Despite being supervised approaches, they are intended to reduce the need for training data as well as provide defect localization to aid the interpretation of results. On the one hand, Convolutional Neural Network is trained using the sliding window processing algorithm to process the images, and on the other hand, a Fully Convolutional Network based approach is employed which is able to process the images as a whole.

These two techniques belong to the block of supervised learning techniques in Figure 1.3, thus they require annotated samples for training. These techniques are thought to be employed in the second stage of the proposed methodology after the anomaly detection is executed. This was thought this way as the anomaly detection model could provide automatic annotations as training samples and avoid the need for manual labeling. Despite conceptualizing the methodology this way, practitioners may have access to enough defective samples and have decided to devote themselves to the task of annotating them. In this case, they will be in the position of directly applying the supervised learning approach.

4.1 CNN and Sliding Window

One possible alternative that was explored to employ the supervised learning approach in a data scarce scenario is mixing the Convolution Neural network for classification with the sliding window algorithm. The idea here consists in splitting the images into several sub-images such that the dataset to train the network could increase and overcome the risk of overfitting. By splitting the image, the network would understand them as independent entities instead of parts belonging to the same cell. Also, is a way of turning an image classification problem (i.e., general evaluation) into a region classification problem (i.e., local evaluation). When giving an entire image to the network, it is expected that it will learn how to focus on the meaningful features in order to classify the entire image in a predefined class (e.g., defective or not). With the sliding window processing instead, the network is guided by telling it which regions are defective and which are not, and thus, which features are important to spot defective cells.

Once the training finishes, the same procedure would be employed to process the test images. In this case, the network would be slid over the images with an overlap between regions, and predict a probability of the defectiveness of each region. At the same time, the predicted probabilities would be accumulated into a final image where the regions that were classified multiple times as defective would present higher values resulting in a heatmap like image.

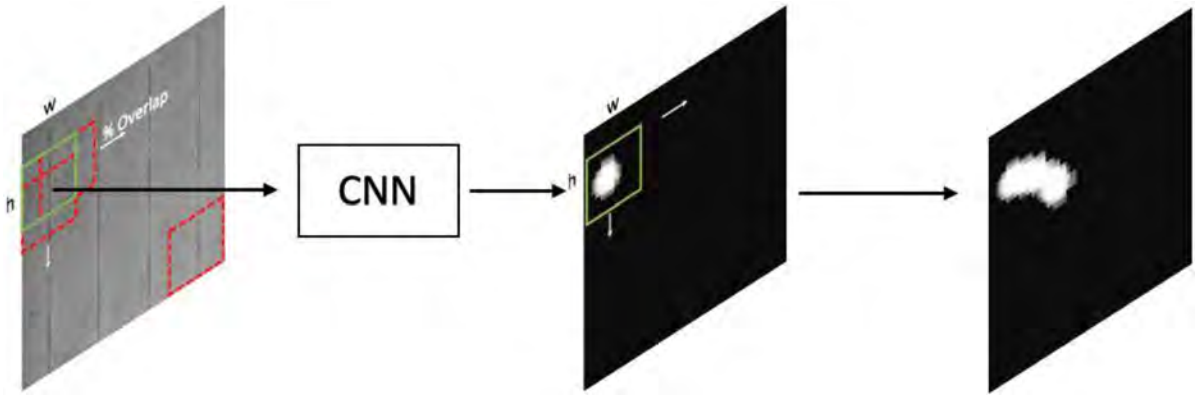


Figure 4.1: Sliding window and CNN based inspection method schema. The trained network is slid across the image, and the output probability is accumulated on the window location resulting in a final heatmap like image.

Sliding window is a broadly used procedure in the field of computer vision, such as, for image filtering (e.g., Gaussian blurring), for local thresholding (e.g., Otsu's algorithm), or in more advanced tasks like object detection [38, 108]. For the sliding window approach, a window (w, h) is defined which is slid from left-to-right and top-to-bottom in the image, and at every location, an operation is performed. For example, in Gaussian blurring, a two dimensional kernel whose entries represent a Gaussian function is convolved with the image (i.e, the dot product between the specified kernel and the region inside the window is computed at every position). In this work, the sliding window procedure was employed to create the training dataset to train the CNN, and then to process the test images and obtain the final heatmap results with the trained CNN. The whole procedure used in this work can be separated into three different phases: 1) Dataset creation, 2) Network training and 3) Trained network deployment.

1) Dataset creation: For the dataset creation, the dataset in Section 2.4 was taken and the sliding window algorithm extracting patches from the original image was applied. In this case, in order to increase a bit the amount of generated image, an overlap ratio of 90% was employed when sliding the window both vertically and horizontally. Also, for

each patch extracted, an additional patch image was created by employing a rotation of 90º to the extracted patch.

In the original images (before the split), the ratio between defective pixels and defect-free pixels is very low as illustrated in Section 2.4. To compensate for such imbalance, two additional images were also created for each patch by rotating the images 180º and 270º based on the number of defective pixels present in the region. If the region had more than 15 defective pixels, this augmentation was executed, otherwise, only the 90º rotation was applied. The pixel counting was performed using the corresponding pixel-wise annotated ground truth images of each cell.

For illustrative purposes: following the described procedure, if 145 defective samples and 397 defect-free samples were taken, a window of size 32×32 pixels, and an overlap ratio of 90%, about 1.9 million defect-free patches and 350,000 defective patches would be obtained. Then, if they were divided into training and testing sets following a common proportion of 90% for training and the remaining for testing, the training set would be composed of about 315,000 defective samples. This will help to overcome the network overfitting issue that could face if the original dataset was directly used. These numbers could be easily increased by adding more augmentation procedures (e.g. affine transformations) or decreased by reducing the overlap ratio or increasing the window size.

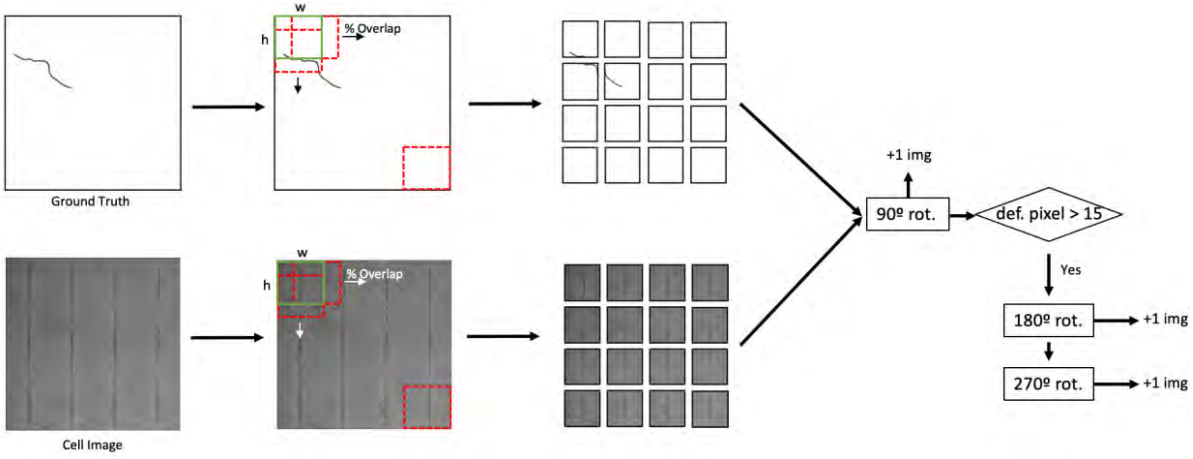


Figure 4.2: Schema of the dataset creation using Sliding window.

2) CNN training: Once the training dataset was created, the patches were separated into train (90%) and test (10%) sets to train and evaluate the neural network, respectively. Regarding the network architecture, several experiments were done with different network architecture configurations inspired by the one in [68]. First, a base architecture (config.

1 in Table 4.2) was defined, trained, and tested. Then, modifications were incorporated to explore different window sizes and check their impact on the final defect detection rates and defect location results. The base architecture was composed of 3 Convolutional Layers (ConvL) and 3 Fully Connected (FC) layers, with some additional layers in between aiming to make the model more robust. The additional layers were Gaussian noise (it adds some noise to the pixels in the images defined by a Normal distribution), Batch Normalization, Max-pooling (MaxPool) layers between the ConvL, and Dropout layers between the FC layers. The experiments with the window sizes consisted in doubling the size for each of the following experiments. This caused the final feature maps before the FC layers to also double in size. In order to keep the size at this point the same for all the networks ($4 \times 4 \times 32$), more layers were included after the final Convolutional blocks. The building blocks used for the construction of the architectures are presented in Table 4.1, and their combination in the different network configurations is shown in Table 4.2.

Conv. Block	**FC block**
gauss. noise 0.1	FC 512
Conv 32, 5×5	dropout 0.3
BN	FC 256
maxPool 2×2	dropout 0.5
–	FC 64
–	FC 1 Sigmoid

Table 4.1: Base block used to construct the different architectures.

Conf.1	**Conf. 2**	**Conf. 3**	**Conf. 4**
input 32×32	input 64×64	input 128×128	input 256×256
3 × Conv. block	4 × Conv. block	5 × Conv. block	6 × Conv. block
FC block	FC block	FC block	FC block

Table 4.2: CNN architecture configurations.

All networks were trained during 50 epochs, with a batch size of 256 images, with RMSprop [49] as the optimization algorithm, and binary cross-entropy in Equation 4.1 as the loss function. The batch size was set to 256 after several short tests, starting with a value of 16 and increasing it by two until there was no improvement in the results. With respect to the optimizer, both Adam and RMSprop optimizers were tested as the most common ones seen in other binary classification problems. After some experiments, RMSprop was selected as it gave slightly better results.

$$\mathcal{L} = \frac{1}{N}\sum_{i}^{N} -y_i \cdot log(p_i) - (1 - y_i) \cdot log(1 - p_i) \tag{4.1}$$

where p is a scalar prediction for each image (patch from the original image) and y the ground truth for each image.

3) Network deployment: The final step in this method consists in employing the sliding window procedure used to generate the training dataset, but now sliding the trained network such that at each step the region is processed by it. The models were designed to output a probability value p about whether the region is defective or not, where the value p is the probability that the area is defective and the value $1 - p$ is the probability that the area is not defective. To build the final result, a new image of the same size as the test image was created. When the network processed a region, the output of the corresponding probability was added at the same position in the new heatmap image. In this case, the output probability was added to every pixel at the region in the heatmap image instead of just the center pixel. Like in the dataset creation, the images were processed using an overlap between adjacent regions such that regions that obtained high probability values multiple times will end up with higher values than their neighbor regions revealing the location of the defects. To avoid excessive FP cases, the output probability was accumulated only when it was above 0.6. By accumulating the probabilities, the defects shapes in the heatmap presented blurred edges. In order to obtain sharper shapes, the images were thresholded.

4.1.1 Experiments and Results

The training and testing of the networks in Table 4.2 was performed on the Polycrystalline dataset distributed as in Table 4.3 and with the first Docker container described in Section 2.5 with CUDA 10, Tensorflow 1.14, and Titan Xp GPU.

The results from these experiments were evaluated quantitatively at image level using Recall and Precision metrics, and also qualitatively with regard to the segmentation.

As the results consisted of heatmaps like images, a thresholding operation was needed to evaluate whether each image as defective or not. After applying the threshold, if there were 15 defective pixels, the resulting image was evaluated as defective, otherwise, it was classified as defect-free. Based on this post-processing, the results were evaluated against the ground truth obtaining the metrics in Table 4.4.

	Train	Test
Defect-free	371	26
Defective	128	17

Table 4.3: Dataset distribution used for the sliding window experiment

	Recall	Precision
Configuration 32×32	89	65
Configuration 64×64	94	65
Configuration 128×128	92	85
Configuration 256×256	89	65

Table 4.4: Results of different configurations at full image level.

The second configuration that corresponds to the network with a window size of 64×64 pixels was the configuration that obtained the highest Recall value that represents the capability of detecting the existing defective samples in the test set. However, the third configuration with the window size of 128×128 pixels obtained a much higher Precision rate with a similar Recall rate. Overall, the best configuration for detecting the defects can be considered the third configuration.

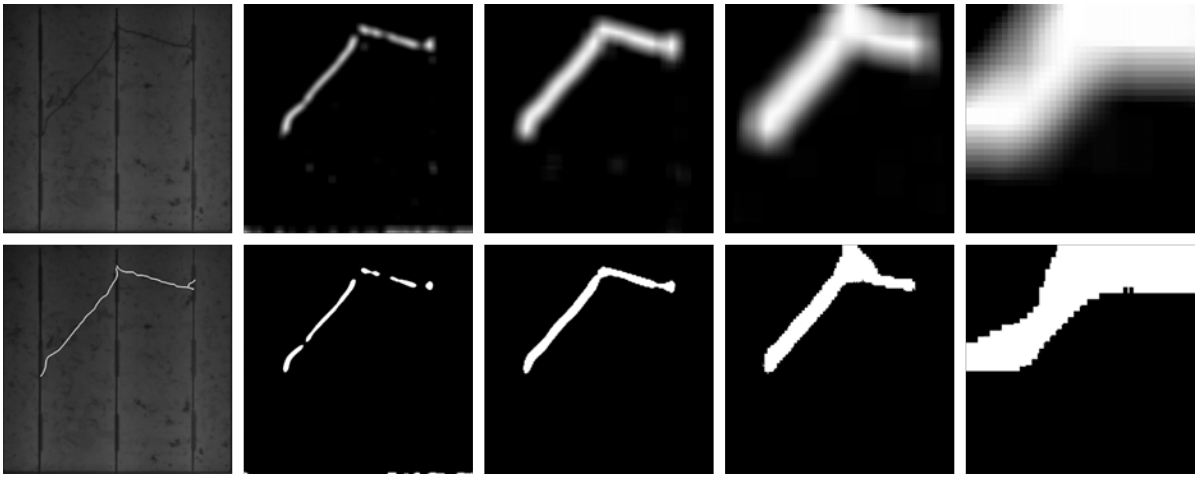


Figure 4.3: On the left, the cell image and the same image but with the defect highlighted in white. Then, from left-to-right the results from 32, 64, 128, and 256 window size networks with an overlap of 90%, and top-to-bottom, the same results but with a threshold applied.

Nonetheless, if both the image level and the pixel level results are analyzed together, it could be said that the second configuration was the model with the best results. While the third configuration was able to segment the entire defect and the predictions kept the underneath shape of the defect, the second configuration was able to obtain segmentation results that were closer to the defect's shape. It should also be mentioned that the first configuration that used a window size of 32×32 pixels was the configuration that was able to get closer to the defect borders even though certain areas remained unsegmented and certain false positives appeared at the bottom of the cell.

Regarding the execution time to process an image, as it was anticipated, the bigger the window size was, the less time was required to process the images. The time that the four models achieved to process an entire image is reflected in Table 4.5. The fourth configuration with the window size of 256×256 pixels was the configuration that was closest to meeting the stipulated time for processing a cell during the real production scenario (half a second at most), however, this was in sacrifice of precise segmentation results.

Model	Time (s)
Configuration 32×32	11-12
Configuration 64×64	5
Configuration 128×128	3
Configuration 256×256	0.9

Table 4.5: Execution time of the different configurations.

The experiments show that the sliding window approach is a way to increase a dataset, it allows one to obtain a segmentation like image using a small CNN network designed classification, and it only requires a single GPU although it can also be run on a CPU. Nonetheless, the networks take much time to process each image which makes the approach unable to match the required production speed. An option to reduce the computation time is to reduce the overlap ratio such that fewer steps would be required to process the images. However, when reducing the overlap for each configuration so the execution time is at most 0.5 seconds, the segmentation accuracy will drop as can be observed in Figure 4.4.

In order to reduce the execution time, but still, obtain accurate location results, the next section will describe how in the following work the sliding window approach was

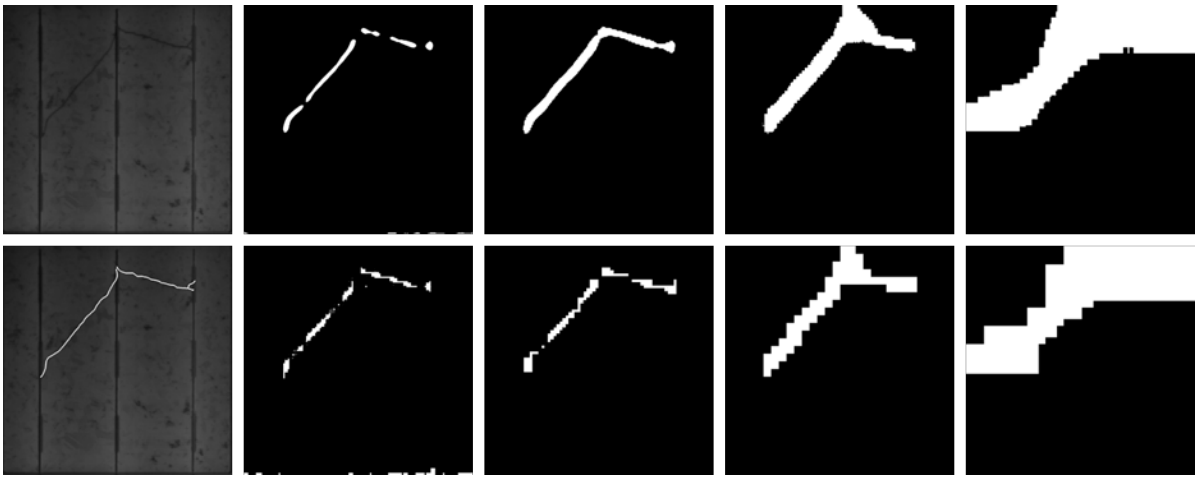


Figure 4.4: On the left, the cell image and the same image but with the defect highlighted in white. Then, from left to right the results from 32, 64, 128, and 256 window size networks with an overlap of 90%, and top to bottom, the same results but with lower overlap (i.e., 40%).

substituted with a Fully Convolutional Network based approach. Instead of processing the images by regions, the FCN is executed by taking the whole image as the input, thus a precise segmentation results can be obtained without requiring a multi-step processing schema.

4.2 FCN based Segmentation

In the previous section, how the sliding window procedure and a CNN designed for classification can be used to detect and approximate the location of the defects in the cells has been described. The main drawback of the method was the need to execute the network multiple times to produce accurate segmentation results. A trivial way to shorten the processing time was to reduce the overlapping ratio of the sliding window, however, this supposed less accurate results. This section will describe how processing time can be significantly reduced and still obtain accurate segmentation results using Fully Convolutional Networks.

The first end-to-end trainable Fully Convolutional Network to obtain dense prediction equal to input size was proposed by [66]. In this work, they substituted the FC layers of a "classic" CNN like the one seen above with deconvolution layers that acted as bi-linear interpolation operations. In this way, the last feature maps in the network

will get upsampled to recover the initial input size allowing the network to learn dense predictions.

In the last years, several end-to-end trainable FCN architectures have been proposed for dense predictions. Among these architectures, the U-net [81] network was selected, which was originally proposed for medical image segmentation (neural structures in electron microscopy stacks). In addition to the similarity between the images in the paper and our case (both grayscale images), the results in the paper show that the network yielded precise segmentation results using few annotated training data. In addition, there was some experience working with this network on another industrial dataset with quite promising results [69].

U-net is composed of two sequential parts: an encoder and a decoder. The encoder follows a similar structure to the CNN in the previous section. With consecutive blocks composed of two Convolutional Layers followed by a MaxPool and a BN layer, it focuses on extracting deep features from the data downsizing the feature maps to half their at every step. After a sequence of 4 downsampling blocks, a bottleneck block adds depth to the network to extract and force the network to learn more meaningful features. Then, the decoder transforms the extracted low-resolution features into a final segmentation map of the input size by successive upsampling steps. The upsampling steps are configured as the inverse of the downsampling blocks, substituting the pooling operators with upsampling operators, which can be defined to be as static bi-linear interpolation or learnable deconvolution operations. Additionally, at every downsampling block, a skip connection is established to fuse the extracted features with the corresponding upsample block with the feature resolution. These connections allow the network to recover feature location information that is lost during the downsample steps at the encoder. Figure 4.5 shows a brief schema of what U-net network architecture looks like.

4.2.1 Experiments and Results

The first experiment in this section consisted in just substituting the network architecture and the image processing scheme from the work in the previous section with U-net. After this, two additional experiments were also done regarding 1) the reduction of the number of layers from the original U-net architecture and 2) the usage of pre-trained weights as an initialization method.

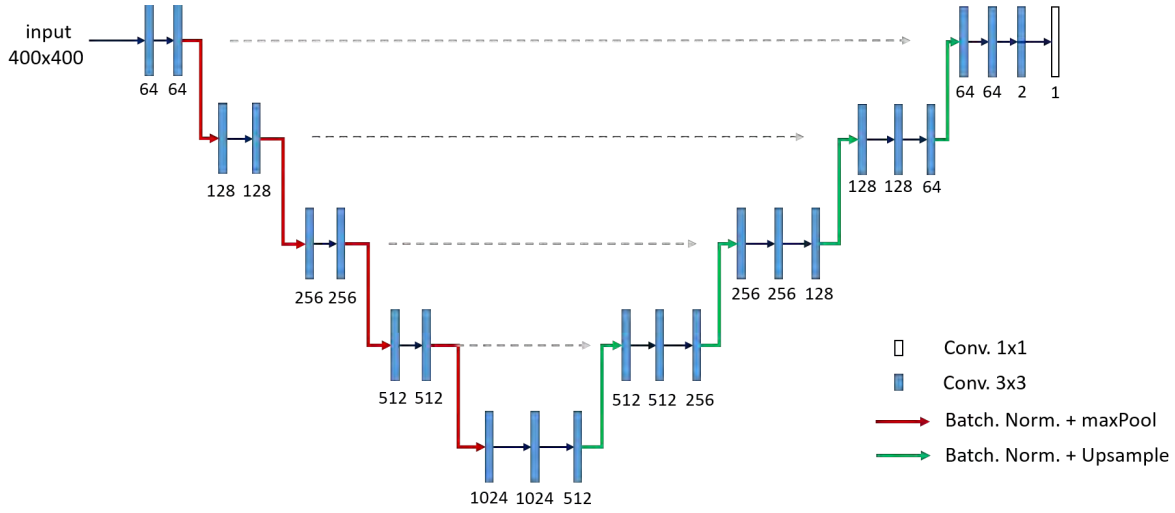


Figure 4.5: Schema of U-net architecture.

Regarding the reduction of the layers, the experiment aimed to check if the network could obtain similar detection rates as the original network even if some features extracted from the deeper layers were removed. As the images go down in the encoding part, they are reduced in dimension which might make information of smaller defects be lost from the feature maps and thus be difficult to detect. On the other hand, when reducing the number of layers in the network, the overall complexity of the network will also be reduced, alleviating to some extent the memory usage both during execution and model storage. The reduction consisted in removing the last three blocks in the encoder parts and their counterpart blocks in the decoder part, and then, connecting the deepest layers to make the network have the encoder-decoder shape like in the original architecture.

In the case of the experiment with pre-trained weights, it will serve to check if it could help to improve the results. Section 6.1 briefly describes how this Transfer Learning is applied, but in short, it consists of using the weights of a network already trained as initialization of the weights of a network to be trained. For more extended information refer to the survey in [104]. The weights were from VGG16 [87] pre-trained on Imagenet [33] and were used in the encoding part of the network.

All the experiments were done using the same container with the Titan Xp GPU and Tensorflow 1.14 as in the experiment in the previous section. In the case of the dataset, the training samples were just limited to the defective samples in Table 4.3. After several experimental training using both defect-free and defective samples and different loss functions, it was observed that it was better to stick to just the defective samples as training samples. When using both types of samples, the network was biased towards the

defect-free class yielding blank segmentation outputs. In other words, for the network, all the cells were always defect-free cells. Instead, when using only defective samples the defects in the test images started to be segmented. As mentioned before, there is a great unbalance between defective and defect-free pixels in the images. In the case of the sliding window approach, the unbalance did not have such a negative impact on the results as images were split into patches and the patches with defective pixels were augmented with rotation operations. In this way, the network could see more defective samples during training which compensated for the unbalance. In the case of U-net, even if image augmentation techniques were applied, there will still be a great unbalance between defective and defect-free pixels as in the original dataset. The only way to alleviate the problem was to remove the defect-free samples and focus on the defective cells by employing rotation and flipping like morphological operations during the training. In contrast to the previous experiment, the images were required to be downsized to a resolution of 400x400 pixels in order to fit all the trainable parameters of U-net in memory during training.

All the U-net based networks were trained to maximize the Dice Coefficient in Equation 4.2 between the ground truth and prediction. The training was performed during 500 epochs with a batch size of 4 images and Adam [54] as the optimization algorithm. The value of the batch size was selected along with the input image size. A batch of 4 was the highest value that the memory could support with a relatively large image.

$$L_{dice} = -\frac{2 \cdot \sum_j p_j g_j}{\sum_j p_j + g_j} \tag{4.2}$$

where $p \in [0, 1]$ is the network output, and $g \in \{0, 1\}$ is the ground truth.

After training models, they were evaluated on the test samples obtaining the results shown in Table 4.6. Note that the results from the Configuration 2 (sliding window of size 64×64 pixels) are different from the ones reported in Table 4.4.

Overall, all U-net based networks were able to detect most of the defective samples, however, the sliding window based approach still surpassed these models in detecting defective samples. Despite that, configuration 2 network mistakenly segmented areas in defect-free samples leading to lower Precision and Specificity rates, i.e., higher False Positive cases and lower True Positive cases than the original U-net.

In the case of the experiments with the U-net based networks, the reduced version of U-net performed similarly to the sliding window approach. It segmented more areas

	Specificity	Recall	Precision
64×64 Conf. 2	73	100	69.5
U-net	88.4	87.5	82.3
Reduced U-net	38	93.7	48.3
Pre-trained U-net	80	81.2	72.2

Table 4.6: Image level results from U-net and sliding window experiment.

as defective, but also incurring in more False Positive cases. It even obtained worse Precision results than the sliding window approach. It seems that reducing layers made the network less capable of learning features that differentiate crystal grains or other structures like the buses from defects. In the case of U-net with pre-trained weights, it did not help much in improving the performance if compared with U-net trained from scratch, but it either was harmed badly.

In addition to the quantitative results, Figures 4.6, 4.7 and 4.8 show some representative samples of the segmentation results that may ease the interpretation of the ratios in the previous table.

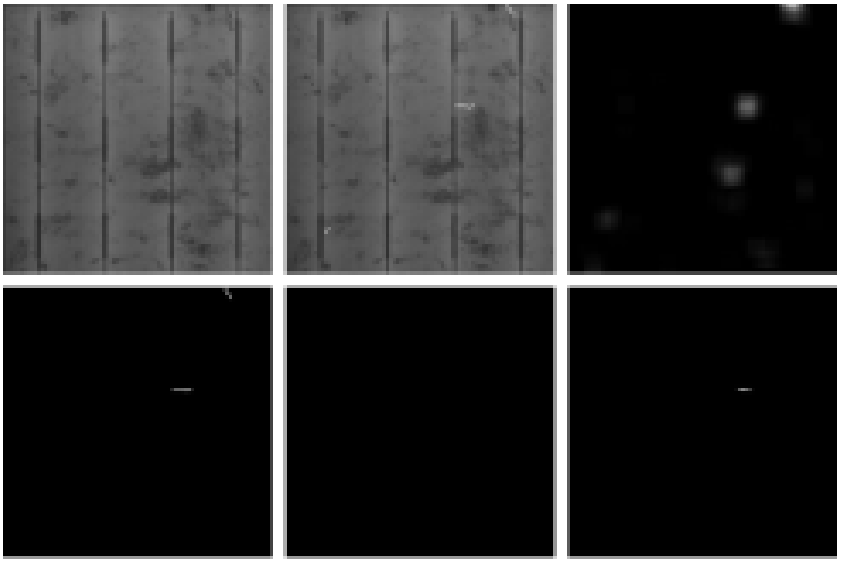


Figure 4.6: Results on severe defects with U-net. In the following order: the cell image, the cell image with the defect highlighted, the results using the sliding window approach with a window size of 64×64, the results using original U-net architecture, the results with the reduced version of U-net, and the results using the original U-net architecture with pre-trained weights.

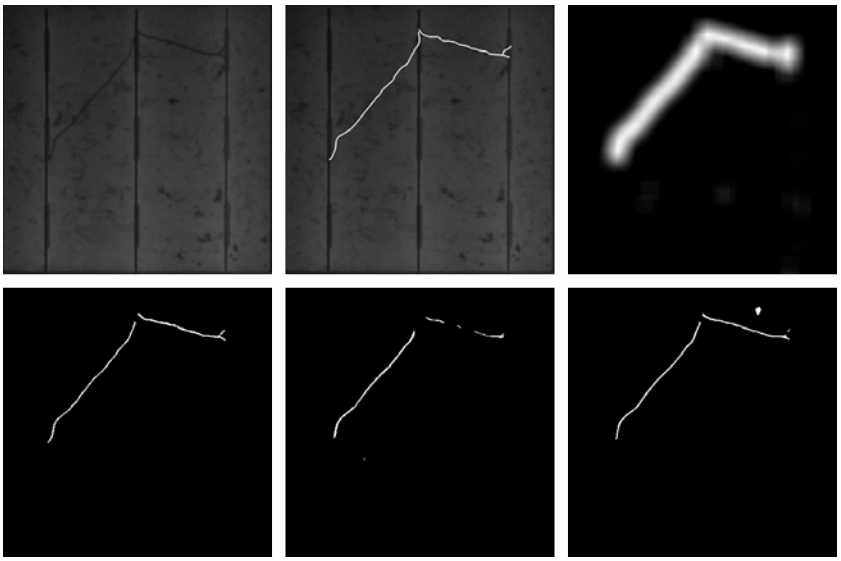


Figure 4.7: Results on light defects with U-net. In the following order: the cell image, the cell image with the defect highlighted, the results using the sliding window approach with a window size of 64×64, the results using original U-net architecture, the results with the reduced version of U-net, and the results using the original U-net architecture with pre-trained weights.

As expected, the networks performed better when the size of the defect was larger. All the networks were able to segment with relatively high accuracy big defects like the Crack in Figure 4.6 but struggle with Microcracks like the one in Figure 4.7. In particular, none of the networks was able to detect the smallest microcrack on the bottom left of the cell in Figure 4.7. Note that this small defect is around 0.75 cm in length, which can be very hard to be detected regardless of the technique being used. As for the segmentation accuracy, it is clearly visible that the sliding window approach led to blurrier results than the U-net based experiments. It is true that the CNN and the sliding window were not designed specifically for segmentation but for classification, nonetheless U-net designed for segmentation is more suitable for cases where the goal is to obtain an accurate location of defects.

With respect to the execution time, U-net reduced the time from the 5 seconds per cell obtained with the sliding window approach to 0.07s per cell. Even if U-net is a more complex network with more layers than the CNN used with the sliding window, the final segmentation results are obtained after only one forward pass instead of the previous multiple executions. Even more, if the network size is reduced as in the case of the reduced U-net which required 0.048s per image.

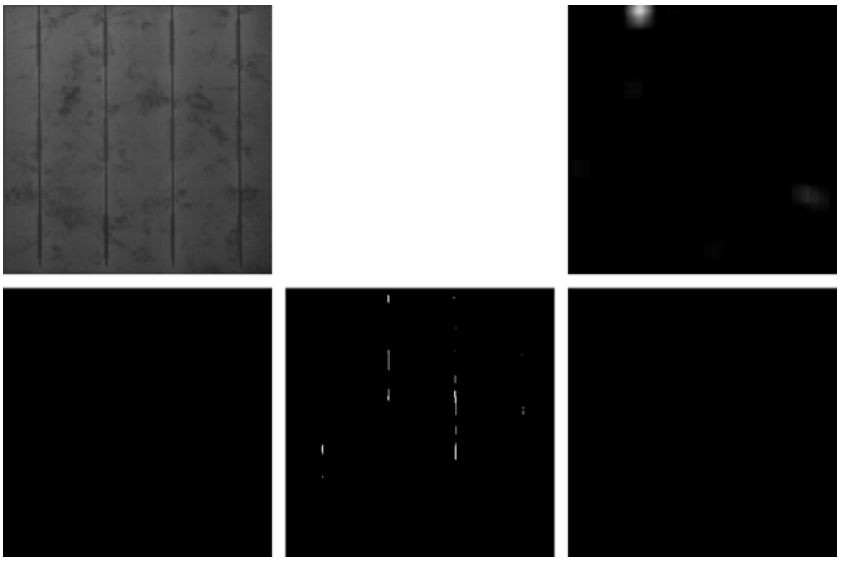


Figure 4.8: Results on a defect-free sample with U-net. In the following order: the cell image, the results using the sliding window approach with a window size of 64×64, the results using original U-net architecture, the results with the reduced version of U-net, and the results using the original U-net architecture with pre-trained weights.

4.3 Concluding remarks

In this chapter, two possible supervised alternatives that employ a few defective data samples and with which the location of defects in the cells can be obtained have been explored.

On the one hand, a sliding window approach in conjunction with a CNN for classification has been explored. With this approach, the images in the datasets are not analyzed as a whole but in patches. This way of processing the images can be understood as a way of augmenting the dataset that can help overcome the lack of defective training samples. On the other hand, a Fully Convolution Network based approach has been explored aiming to speed up the processing time shown by the sliding window approach as well as improve the defect location accuracy.

Overall, the results from the experiments have shown that U-net based approach yielded more precise segmentation results than the sliding window approach, reducing at the same time the execution time per image. However, even though it was more precise at segmenting defects in the cells, the image level detection rates experienced some drop with respect to the results from the sliding window approach.

Supervised training

The supervised learning approaches are the way to go when there are enough defective samples and annotations available. However, real production lines may still present certain limitations with respect to collecting enough defective samples for these supervised methods to be used. The next chapter will explain how another approach can overcome such limitations and still obtain accurate defect location in the cells.

Chapter 5

Anomaly detection

In industrial quality inspection environments, there are different peculiarities that must be taken into account when applying DL-based solutions. These sometimes are difficult to gather, making it hard to generate large enough datasets with representative images of the different characteristics of interest for appropriate training. In addition, the manual labeling of each of the examples must be done, which is usually an arduous task that takes plenty of time and resources. In new industrial processes, there are no defective data samples from the beginning, so it would be necessary to wait a long time to be able to have enough data to train DL models capable of identifying the faults that may appear.

Fortunately, there are ways that take advantage of defect-free samples which are more accessible even if the process is new. This chapter will describe how these defect-free samples can be used to train a neural network using an anomaly detection approach and obtain a model that can classify defective and non-defective cells as well as locate the defects.

5.1 Anomaly Detection

Anomaly detection refers to the problem of finding patterns in the data that differ from what is considered normal. In an industrial quality inspection setup, anomaly detection can be translated as finding defective patterns during production using defect-free samples

as training data. For this purpose, in the context of Deep Learning neural networks are trained using just normal data such that they learn the probabilistic distribution of what is normal [18]. Then, at inference it is expected that it will only be able to handle defect-free samples, allowing in this way to identify those samples that do not follow the learned distribution, i.e., anomalous or defective samples.

Within the solar cell inspection domain, this approach has been employed in [77] and [76], and lately also in [73] through the usage of Autoencoders (AE), linear AE in the case of the first two works and convolutional AE in the last work. In these works, AEs are trained to encode and reconstruct defect-free samples, so later on it will make the network unable to reconstruct defective patterns. In the case of defect-free samples, the network will output a high fidelity version of the input image, whereas when processing a defective cell, it will output a defect-free version of the input sample. By subtracting the defect-free version from the defective sample, deviations will highlight anomalous patterns in the image.

This same approach has also been employed in other domains but in this case, using Generative Adversarial Networks (GAN) [43] which have shown remarkable capabilities at generating realistic images, and also better anomaly detection rates than AEs [84, 12].

Taking into account the results obtained using GANs, this section will describe the experiments performed with this type of network for anomaly detection in the context of solar cells, specifically the experiments and adaptation performed using the f-AnoGAN network [84]. This network is rooted in the original GAN [43], however, is the result of successive improvements that have been done with regard to the network training stability. Thus, before starting with the description of the particular network, it is considered appropriate to briefly introduce which and how were these improvements incorporated in the original GAN to clarify the reason behind the use of f-AnoGAN.

GANs are a type of generative model composed of a generator G and a discriminator D that are trained in an adversarial manner. The problem is formulated as a zero-sum mix max game Equation 5.1, where G learns to generate samples from an input noise vector z sampled from latent space Z to fool the discriminator, while D attempts to distinguish the generated samples from real samples x drawn from the training set.

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x\sim\mathbb{P}_r}[logD(x)] + \mathop{\mathbb{E}}_{\tilde{x}\sim\mathbb{P}_g}[log(1-D(\tilde{x})] \tag{5.1}$$

where P_r is the training data distribution, P_g is the model distribution defined by $\tilde{x} = G(z)$. As it is a zero-sum game, the training converges at the point known as Nash Equilibrium, where both the generator and the discriminator are good at their corresponding tasks. A successful training makes the generator implicitly minimize the divergence given by Jensen–Shannon Divergence (JS) measure between the generated data distribution $p(z)$ and the training data distribution $p(x)$. In other words, it has learned the probabilistic distribution of the training data.

The minimization of the JS divergence derives into unstable training when both distributions are very different from each other, finally leading to a mode collapse or vanishing gradients [4].

Mode collapse happens when the Generator is only able to generate one kind of sample. During the training, it learns that generating that specific sample is a way of minimizing the loss function (fool the discriminator) and does not try to generate different looking samples as it does not want to get a higher error value. As a consequence, the training ends up with a poor generator that has learned a narrow version of the probabilistic distribution of the training data.

The problem of vanishing gradients instead is a consequence of the nature of the training itself. At the beginning of the training, both the generator and the discriminator are not optimized for their tasks, however, for the discriminator, it is easier to discern a generated sample from a real one than for the generator to learn to generate real looking samples. Due to the form of the JS divergence based loss, the discriminator can end up converging while the generator is still training. This causes the discriminator to stop reporting gradients on which the generator is being trained leading to a model failure as the learning process can no longer continue.

In order to solve the training instability, in [5] they proposed to use the Wasserstein distance or Earth-Mover's distance (EM) Equation 5.2 as the loss function which yielded the Wasserstein GAN (WGAN) architecture. The loss measures the distance between two probabilistic distributions in form of the "cost" γ that will take to transform the generated distribution P_g into the real distribution P_r.

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \tag{5.2}$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ is the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Unlike JS divergence, [5] proved that Wasserstein distance is continuous

and provides usable gradients everywhere, thus avoiding the vanishing gradient problem encountered in the original GAN proposal. The original form of the Wasserstein distance is intractable as it requires exhausting all joint distributions in $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ to find the optimum $\mathbb{P}_g$ that minimize the distance, so it becomes not applicable as a loss function. In order to overcome this limitation, the Kantorovich-duality was employed to simplify the equation turning it into the loss function in Equation 5.3.

$$\min_{G} \max_{D \in f} = \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r}[D(x)] - \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] \tag{5.3}$$

where f is a set of 1-Lipschitz functions, and $\mathbb{P}_r$ and $\mathbb{P}_g$ are again the training data distribution and the model distribution respectively. In this case, under the optimal discriminator, renamed to "critic" as it does not classify now, there will be gradients for the generator to train. By minimizing the loss function with respect to the generator parameters, the Wasserstein distance between $\mathbb{P}_r$ and $\mathbb{P}_g$ will be minimized.

Nevertheless, the weights in the critic need to lie in a compact space $[-c, c]$ that fulfill the 1-Lipschitz constraint. To enforce that condition, the weights were clipped after the gradient update at an arbitrarily defined value. However, a wrong clipping value can make it harder to optimally train the network, so in a future work [45], they extended the loss with a gradient penalty term (WGAN-gp) Equation 5.4 that dynamically performs the weight clipping enforcing the 1-Lipschitz constraint based on the input.

$$L_{WGAN} = \mathop{\mathbb{E}}_{\tilde{x} \sim \mathbb{P}_g}[D(\tilde{x})] - \mathop{\mathbb{E}}_{x \sim \mathbb{P}_r}[D(x)] + \lambda \mathop{\mathbb{E}}_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[\|\nabla_{\hat{x}}(D(\hat{x})\|_2 - 1)^2] \tag{5.4}$$

where $\tilde{x} = G(z)$, $\hat{x} = \alpha x + (1 - \alpha)\tilde{x}$ with $\alpha \sim U(0, 1)$ and λ is the penalty coefficient.

f-AnoGAN [84] uses the WGAN-gp as base architecture that helps having a stable training while keeping the capability for generating realistic images used for a successful anomaly detection. The network was originally proposed for anomaly detection in the medical domain where it is also difficult to obtain anomalous samples (e.g., diseased samples). The original network design does not contemplate the processing time as a key aspect, which makes the network too slow for its application under real industrial cycle speed (more than a second per image). For this reason, the network has been adapted to reduce the processing time and also improve the detection rates.

5.1.1 Anomaly detection model (f-AnoGAN)

f-AnoGAN is composed of three different sub-networks (a generator G, a discriminator D, and an encoder E) that are trained in two phases.

In the first training phase, the generator and discriminator are trained in an adversarial manner to learn a latent space of normal data variability using just normal data. In this work, defect-free samples are considered as normal data and defective samples as anomalous data.

In the second phase, the encoder is trained to map normal data from the image space to the learned latent space while the Generator and Discriminator are kept unaltered. Once these two phases have finished, the encoder can map test images from the image space to the latent space, and the generator can reconstruct the encoded version of the images from the latent space back to the image space. As the network is trained on normal data, it only learns to encode and reconstruct correctly normal features; thus, when processing anomalous samples, deviations from the reconstructed images can be used for anomaly detection and location.

Phase 1 - WGAN training

The objective of the first phase consists in learning the variability of normal data. For this purpose, the generator and the discriminator are trained following the architecture of WGAN and the gradient penalty based loss in Equation 5.4 to minimize the Wasserstein distance between the real normal data probability distribution P_r, and generator synthesized data probability distribution P_g is minimized.

During the training, the generator is fed with a noise input vector z, sampled from a latent space Z, and tries to learn the mapping from that latent space to the image space X. The synthesized data $G(z)$ should follow as close as possible the real data distribution P_r. Simultaneously, the discriminator is given the generated sample $\tilde{x}$ and the real sample x so it outputs a scalar about how close both distributions are. The training and the components in this phase are illustrated in Figure 5.1.

After the first phase of training, 1) a latent space that represents the variability of the normal data, 2) a generator that can map samples from this latent space to image space, and 3) a discriminator that can detect samples that do not follow the normal data distribution are obtained. However, at this stage, there is no network component that

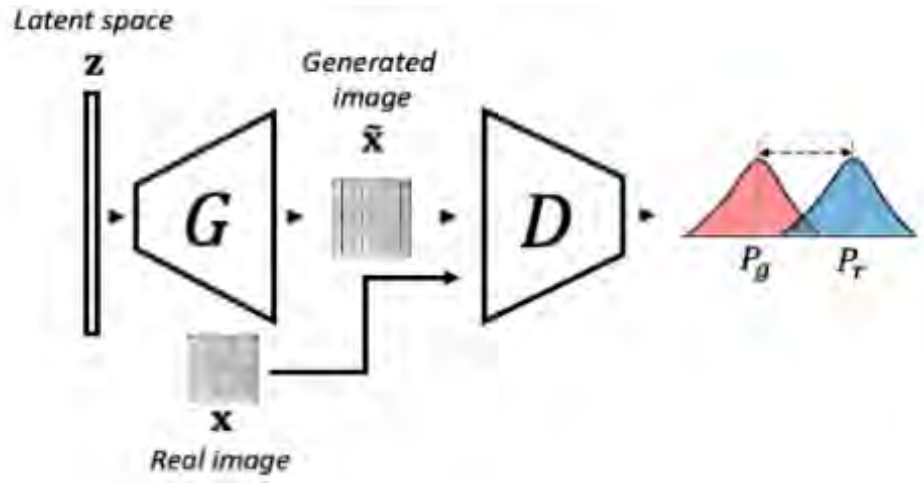


Figure 5.1: The schema of phase 1 of f-AnoGAN training.The Generator takes a vector **z** and tries to generate an image that follows the same distribution of the real data. Then, the Discriminator measures the difference between the generated data distribution and the real data distribution.

can perform the inverse mapping, i.e., from image space to latent space. The next phase will focus on learning this mapping.

Phase 2 - encoder training

In the second training phase illustrated in Figure 5.2, the objective is to make the encoder learn to map a real image to the latent space such that the generator can map it back to the image space. During this stage, both the generator's and the discriminator's weights remain unaltered. This network configuration is denoted as izi in [84]. In this case, the encoder is optimized by minimizing the Mean Square Error (MSE) with respect to the difference between the original image x and the reconstructed one $G(E(x))$. Additionally, the reconstruction error from the izi architecture loss is extended by feature residuals from an intermediate layer in the discriminator, yielding the izi_f architecture.

By taking into account these residuals in the feature space, the reconstruction is improved [84]. The final loss function of izi_f is defined by Equation 5.5:

$$L_{izi_f} = \frac{1}{n}\|x - G(E(x))\|_2 + \frac{k}{n_d}\|f(x) - f(G(E(x)))\|_2 \tag{5.5}$$

where $f(\cdot)$ are discriminator's intermediate layer features, n_d is the dimensionality of the intermediate feature representation, and k is a weighting factor.

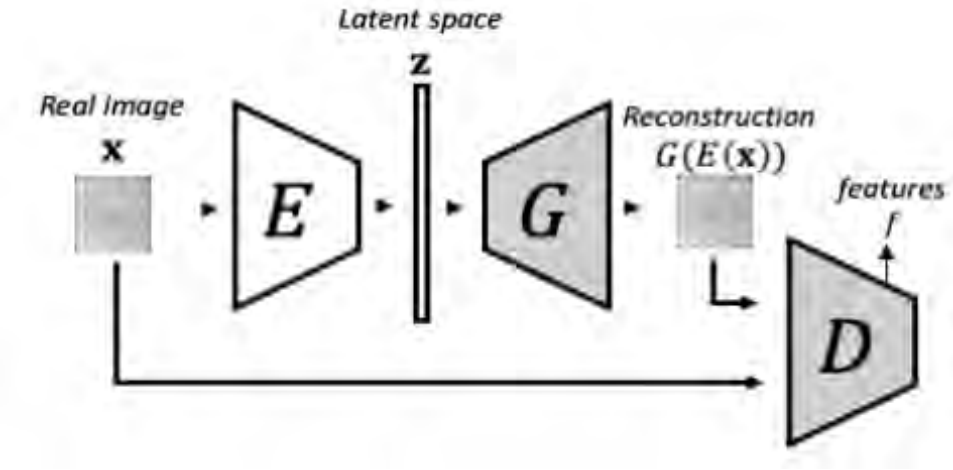


Figure 5.2: The schema of phase 2 of f-AnoGAN training. In the second phase, the Generator and Discriminator are kept unaltered while an Encoder is added and trained to learn to encode the images to the latent space so the Generator can reconstructed them back.

Anomaly detection

Once the training has finished, all the components are fixed and ready to be used for anomaly detection. During the detection, the images are processed as in the encoder training. First, the encoder maps the images to the latent space, and then, the generator maps them back to the image space. Finally, the difference between the reconstructed and the original image defined in Equation 5.6 is used for anomaly detection.

$$A(x) = A_R(x) + k \cdot A_D(x) \tag{5.6}$$

where $A_R(x) = \frac{1}{n}\|x - G(E(x))\|_2$, $A_D(x) = \frac{1}{n_d}\|f(x) - f(G(E(x)))\|_2$ and k is a weighting factor from Equation 5.5.

As mentioned at the beginning of the chapter, in anomaly detection in the context of DL only defect-free cell samples are used for training, therefore the network just learns to reconstruct normal samples. In the case of defect-free samples, the network outputs an image similar to the input image, thus there is not much deviation when subtracting one image from the other. Instead, when processing a defective cell, the output is a defect-free version of the input sample. As a consequence, the deviation between the original and reconstructed images can be used to detect anomalous parts. This behavior is shown in Figure 5.3.

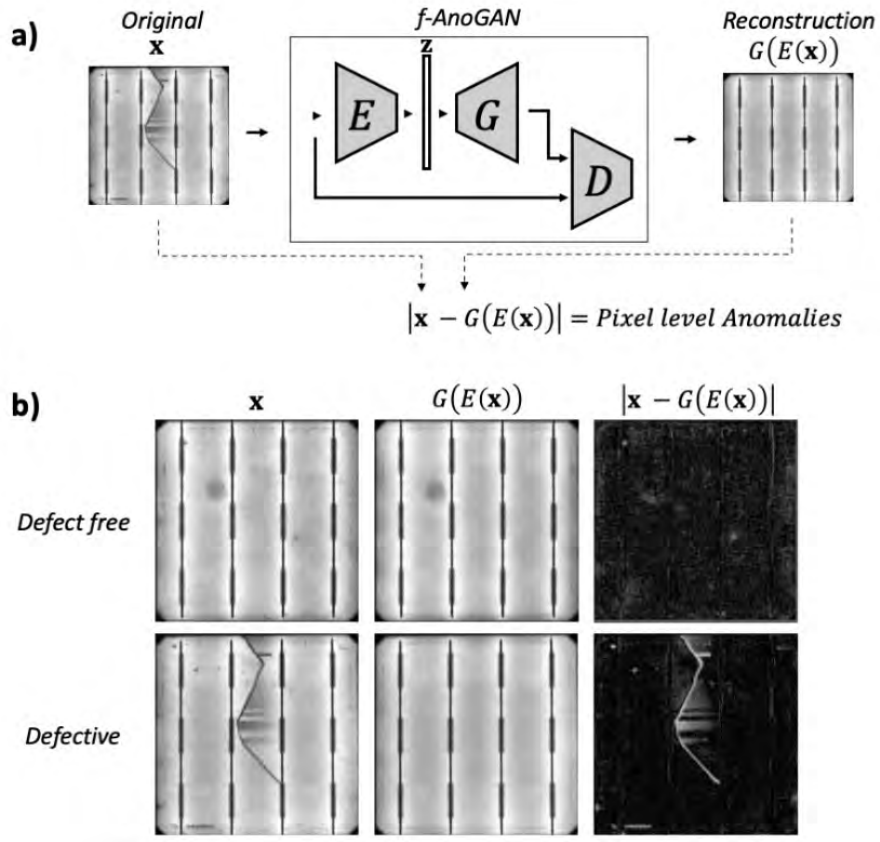


Figure 5.3: Example of anomaly detection with f-AnoGAN. In a) the final structure of the network used for anomaly detection, and in b) some example results obtained when the network processes a defect-free cell and a defective cell.

The absolute value of the pixel-wise difference between the original and the reconstructed image, $|x - G(E(x))|$, is used for pixel-wise anomaly detection. By applying a threshold c, defined in Equation 5.7, to the residuals image obtained from $|x - G(E(x))|$, the binary image $y \in \{0, 1\}$ is obtained.

$$y = \begin{cases} 1, & |x - G(E(x))| \geq c \\ 0, & otherwise. \end{cases} \tag{5.7}$$

5.1.2 Experiments and results

Before experimenting with the network, first, the original f-AnoGAN was employed in our dataset to ensure its applicability in this specific industrial context. Then, two different

modifications were incorporated to improve the results regarding the defect detection rate and processing time.

f-AnoGAN modifications

With f-AnoGAN, the images are processed in patches of size 64×64 pixels, which requires multiple executions of the network, increasing the time to process an entire cell. As a consequence, the network does not meet the industrial production cycle time (under half a second per cell). In order to reduce the inspection time, 1) the encoder input and the generator output layers' dimension was increased. Thus, whole cell images will be processed in a single pass, reducing processing time drastically with respect to the original sliding window approach Figure 5.4.

In addition, 2) the training scheme was also modified. In f-AnoGAN, the generator is frozen during the second training phase in Section 5.1.1; thus, only encoder weights are modified. This can limit the network's capability in terms of reconstructing the input image. In order to maintain a stable training without restricting the reconstruction capability, the generator is also trained at a certain number of the encoder training iterations with a lower learning rate, while keeping the discriminator unaltered. By training the generator, the reconstruction of defect-free samples will improve. Therefore, the deviation between the original and the reconstructed images of normal data will be reduced. Consequently, both the anomaly score and the pixel differences will be lower for defect-free samples, but higher for defective ones; thus, the model's detection rate will improve.

From now on, the original architecture and the architecture with the modifications will be referred to as f-AnoGAN-64 and f-AnoGAN-256 respectively.

The hyperparameters in both models were all kept the same as in [84]: The $\mathbf{z}$ vector was sampled from a Normal distribution and had a size of 128, the value of λ parameter for the gradient penalty was 10, and the value for the weighting factor k in Equation 5.5 was set to 1. The optimization algorithm for the first training phase was Adam [54] and for the second RMSprop [49]. For both models, all the images were rescaled to a range $[-1,1]$ as in the original work [84]. In this way, the pixels in the images will match the range of the Generator output layer activation function (i.e., Tanh), and it will help the network have a stable training [28]. The only hyperparameter that was modified was the batch size for f-AnoGAN-256 training in order to fit the model in memory. This

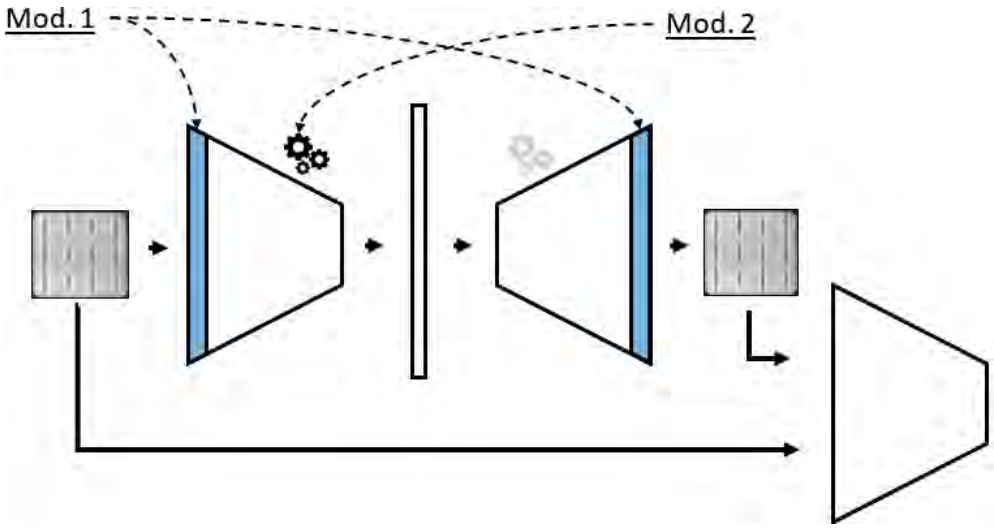


Figure 5.4: Schema of the changes performed in f-AnoGAN architecture.

was due to the increase in trainable parameters resulting from the modification of the architecture. The training took a different number of iterations depending on the phase. Both models required 40,000 iterations in the first phase and 70,000 iterations in the second phase to converge.

Baseline models

In addition to the mentioned models, two Convolutional Deep Autoencoders were also trained. These models were used to establish a base with which the results from the previous two models could be compared. In addition, the results from these two base models served to check if a more simple network architecture could be enough to obtain high defect detection rates in this specific context. Following the two approaches from previous models, one Autoencoder was trained to process the images in patches, and the other Autoencoder was trained to process the images in an image-wise setup. These models will be referred to as AE-64 and AE-256.

Regarding the architectures, both networks are composed of an encoder and a decoder with several convolutional layers. In the case of AE-64, the encoder has two convolutional layers with 64-32 filter distribution, followed by 4 Fully Connected layers of 128 units each, and finally a decoder with the inverted shape of the encoder part. In the case of AE-256, the architecture is two convolutional layers deeper than the AE-64 such that the output dimension before the Fully Connected layers is the same. The filter distribution is 8-16-32-64. After each Fully Connected layer, a dropout layer with a drop rate of 0.25

was set. Both networks were optimized with the MSE loss function and Adam as the optimization algorithm. The AE-64 model training took about 30k iterations with a batch size of 32, and the AE-256 model training took about 6k iterations with a batch size of 8.

Results

The experiments in this section were carried out using the dataset of monocrystalline cells described in Section 2.4. In this case, the network only requires defect-free samples for training, thus, the defect-free samples were separated into the train, validation, and test sets. In addition, in order to perform a quantitative analysis, defective samples were also employed for testing. In the dataset there are certain defects that do not contain a great number of samples to compute representative performance metrics, thus only Cracks, Microcracks, and Finger interruptions were used for the quantitative analysis while the other types were only employed for the qualitative analysis. The dataset distribution used in this part is illustrated in Table 5.1.

	Train	**Val**	**Test**	**Total**
Defect-free	750	373	375	1,498
Defective	-	-	375	375
Crack	-	-	18	-
Microcrack	-	-	240	-
Finger interruptions	-	-	117	-

Table 5.1: Dataset sample distribution for unsupervised part experiments.

The Crack class does not have either a great number of samples, however, this type of defect is one of the most common and important classes of defect to be detected at inspection due to its consequence on the cell's future performance. Because of this, Cracks were also kept in the dataset for the quantitative analysis.

f-AnoGAN-64 and AE-64 were designed to process the images patch-wise. For these cases, each image was split into 256 patches using a sliding window. The final train, validation, and test sets were composed of 192,000, 95,488, and 192,000 images, respectively. For the other networks, the images were resized to the network input size (i.e., 256×256 pixel resolution).

Anomaly detection

To compare the performance between the models, the results from the patch-based models were post-processed. While, in the image-wise models, it was enough to apply a single threshold so as to classify a sample as anomalous or normal, in patch-based models, the errors of all patches belonging to the cell must be taken into account. So, in the latter, the same threshold was applied to every patch, and, if a single patch was evaluated as defective, the entire cell was also evaluated as defective.

Regarding the image-level results, the network modifications had a positive impact on the results. f-AnoGAN-256 was able to detect more defects (higher Recall values) than the original f-AnoGAN-64 without incurring in more False Positive cases (higher Precision and Specificity values). This is particularly visible in Table 5.2 for the case of the finger interruption and microcrack defect classes where all the metrics improved by over 10 points. This improvement can also be appreciated in the ROC curves and the AUC values in Figure 5.5 where the curve reflecting the performance of f-AnoGAN-256 appears closer to the top left corner that represents the perfect classifier, and the AUC value that changed accordingly.

If the results of f-AnoGAN models are compared with the ones from the Autoencoders, it is further underlined that the incorporation of the modifications brought an improvement in defect detection rates. Setting aside the case of finger interruptions, f-AnoGAN-64 obtained worse detection rates than its Autoencoder counterpart (i.e., AE-64) and also AE-256. But, when the proposed changes were incorporated, the obtained results surpassed the ones from the Autoencoders for all the classes and all the metrics, which means higher True Positives cases and lower False Positive cases for all defect classes.

The results in the ROC curves in Figure 5.5 and the metrics in Table 5.2 show that all models could detect all samples with cracks, but they could not detect all samples with microcracks and finger interruptions. This is caused by the fact that cracks are defects that cover a larger area of the cells than finger interruptions or microcracks, therefore having defective pixels that result in a higher anomaly score. The same happens in the case of finger interruptions and microcracks. The first appears in groups of three or more, whereas the latter appears isolated. Because of this, the sum of defective pixels in samples with finger interruptions contributes to higher anomaly scores resulting in higher detection rates.

With respect to the defect location results, Figure 5.6 shows that all the models were able to properly locate the different defect classes. Nevertheless, the segmentation results were more refined in f-AnoGAN-256 and AE-256 models. Although the patch-based

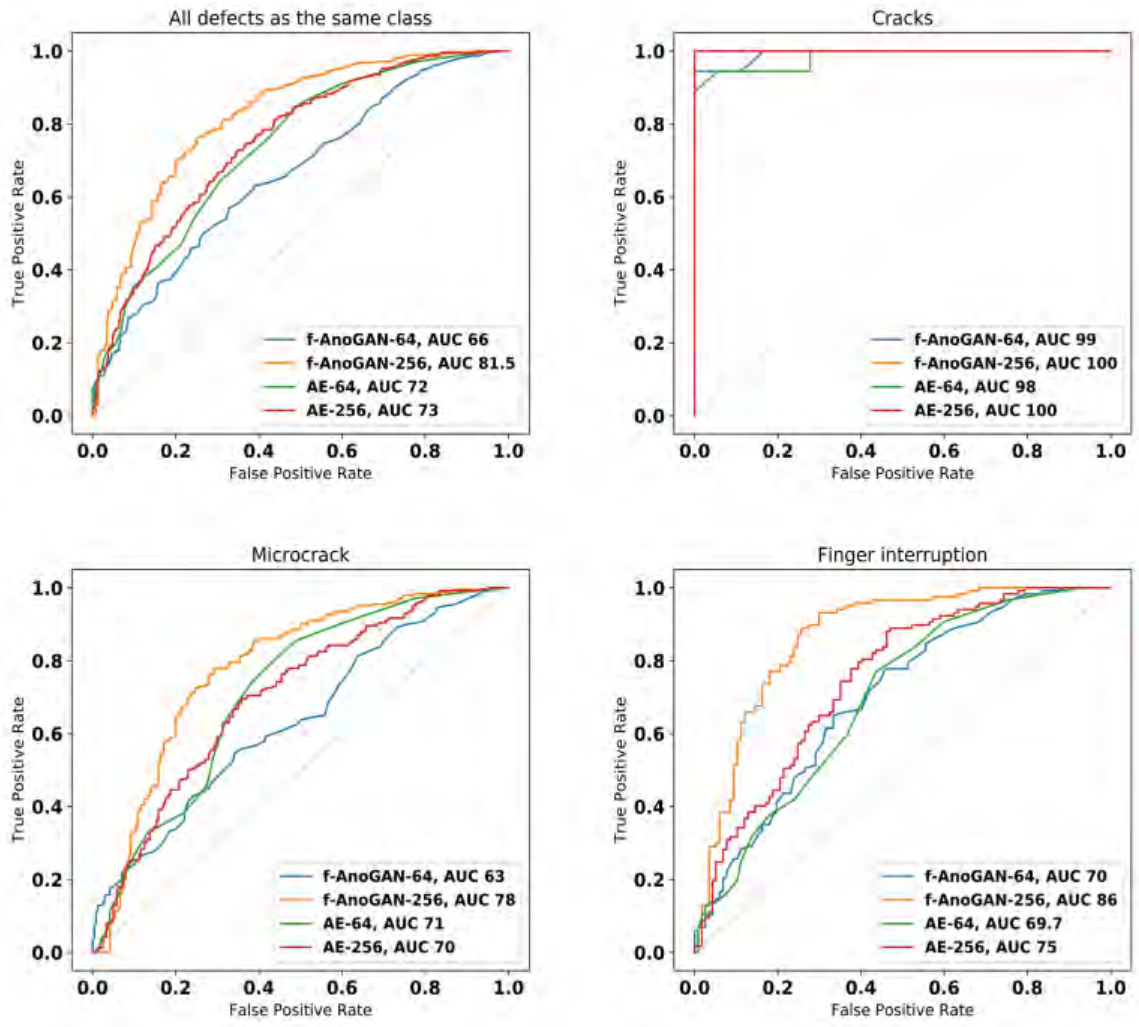


Figure 5.5: ROC curves from the different models in the unsupervised part experiments.

models were able to point out the presence of defective areas, the borders and shape of the predictions were not as accurate as those from the image-wise models.

Also, the patch-based models have more False Positive cases. An example of this behavior is the sample from the second row, where the buses in the cell were mistakenly detected as defects. The same happened on the defect-free samples, where patch-based models classified defect-free areas as defective (e.g., samples seven and eight), whereas the image-wise models obtained clean predictions. Although not illustrated in Figure 5.6, this behavior was shared across several other samples in the test set.

In addition, in Figure 5.6 there are also some results illustrated in regard to samples that contain breaks and bad soldering types of defects (in the bottom-right of the first

	Model	AUC	Precision	Recall	Specificity	f1-score
All test samples						
	f-AnoGAN-64	66	61.3	62.8	61	62
	f-AnoGAN-256	**81.5**	**75**	**78**	**75**	**77**
	AE-64	72	65.6	64	68	65
	AE-256	73	68.4	58	72	63
Cracks						
	f-AnoGAN-64	99	66.7	100	50	80
	f-AnoGAN-256	**100**	**95**	100	**94**	**97**
	AE-64	98	78	100	100	87.7
	AE-256	100	95	100	94	97
Micro						
	f-AnoGAN-64	63	58.7	59	59	58.9
	f-AnoGAN-256	**78**	**73**	**73**	**74**	**73**
	AE-64	71	66.5	63.7	67.9	65
	AE-256	70	66	53	72	59
Finger int.						
	f-AnoGAN-64	70	66	64.9	66.7	65.5
	f-AnoGAN-256	**86**	**78**	**85**	**75**	**81**
	AE-64	69.7	61.9	59.8	63	60.8
	AE-256	75	69	63	71	66

Table 5.2: The results of anomaly detection at the image-level. Precision tells how accurate the classifier is when classifying a sample as defective. Recall tells how many samples have been correctly classified as defective from all defective samples. Specificity describes how many defect-free samples have been correctly classified as defect-free samples. The F1-score is the harmonic mean of the Precision and Recall. In all metrics, the higher the value, the better the classifier is.

row sample and on the right in the third row sample) that were put aside from the quantitative analysis due to the lack of available samples. In the case of the break, image-wise models were able to output a relatively precise segmentation. The AE-64 model results indicated the defect location; however, they did not have much precision. Instead, in the case of f-AnoGAN-64, it can be noticed that, at the defect location, there is a certain anomalous pattern but very vaguely segmented. Regarding the bad soldering, f-AnoGAN-256 was the only model that presented a reasonable segmentation result.

Regarding the processing time, Table 5.3 shows how the architecture modification made f-AnoGAN able to reduce the time required to process each cell. While patch-based models required more than half a second to process the cells (maximum stipulated time

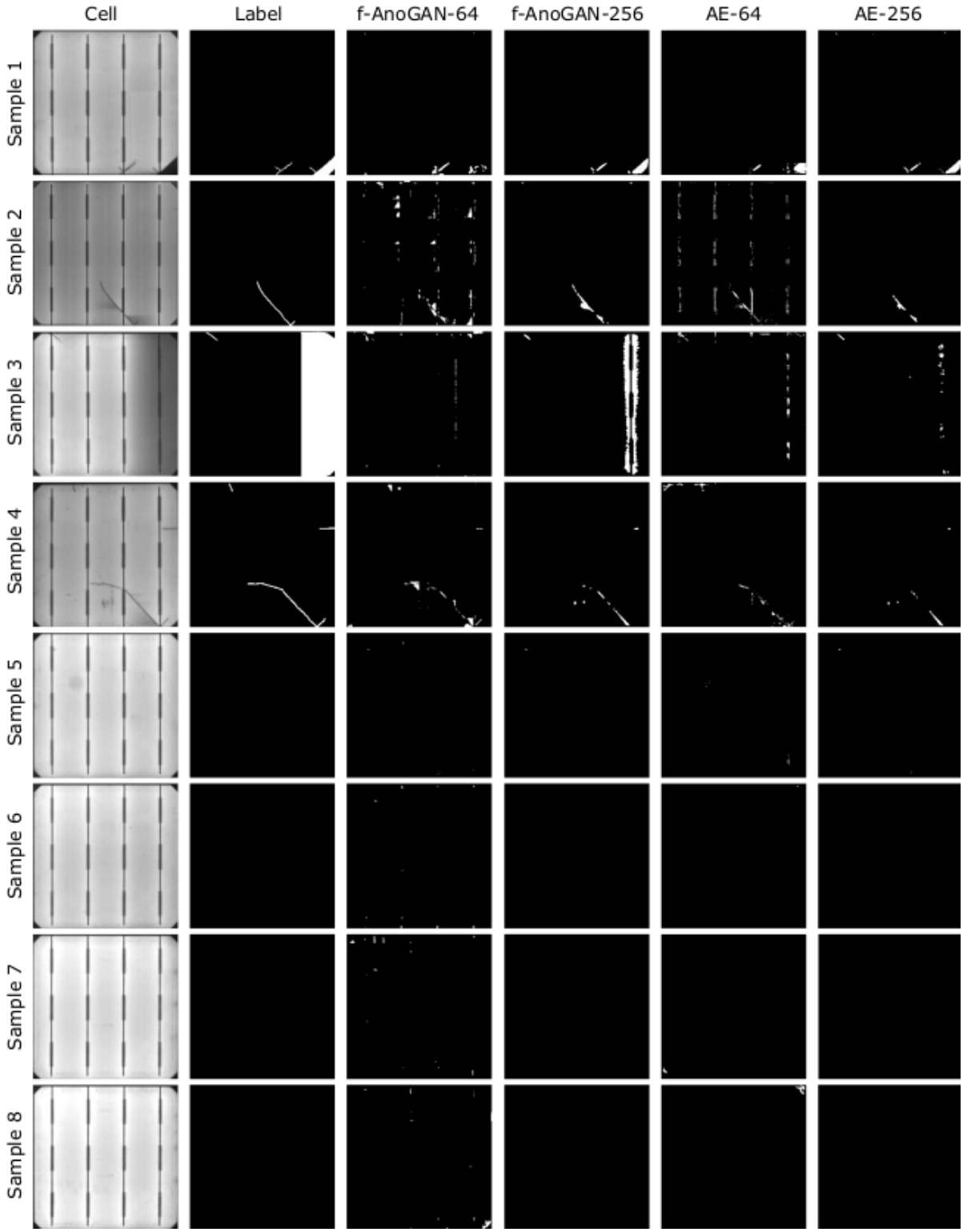


Figure 5.6: Defect localization results from each model.

per cell), f-AnoGAN-256 and AE-256 required only 0.05 and 0.02 s, respectively, to process each cell.

Model	time per patch	time per image
f-AnoGAN-64	0.02s	5.12s
f-AnoGAN-256	-	**0.05s**
AE-64	0.012s	3.07s
AE-256	-	**0.02s**

Table 5.3: Time required to process a cell for each model.

Additional experiments

In addition to the experiments in the previous section, supplementary tests to check the individual influence of each of the modifications incorporated into the network on the detection rates were also carried out. Four different network configurations were trained: the original network (f-AnoGAN-64), the network with the input and output layer size modification (f-AnoGAN-256), and the same two architectures but trained with the modified training scheme modification (f-AnoGAN-64* and f-AnoGAN-256*). Note that the configuration f-AnoGAN-256* in this section is the same as the configuration f-AnoGAN-256 with both modifications that is illustrated in the results in previous sections. The detection rates from each of the configurations are presented in the charts in Figure 5.8. The detection rates are shown as if all the defects belonged to the same class, and taking each of the defect types separately.

And in Figure 5.8, the pixel level results from these additional experiments are reported. To ease the comparison, the image to describe the pixel level results in the previous section is reused and just the results from these latter experiments are incorporated.

As can be appreciated, the input and output layer dimension modification did not improve the results much (f-AnoGAN-64 vs f-AnoGAN-256). The Recall value was worse than with the original network in all the cases. In the case of Precision and Specificity instead, the results were better. The only clear positive side of this modification was the reduction of processing time, which was reduced from 5s/image to 0.05s/image making the network able to meet the established maximum inspection time.

With respect to the training scheme modification (f-AnoGAN vs f-AnoGAN*), it can be concluded that it was overall beneficial for the defect detection capability of the networks

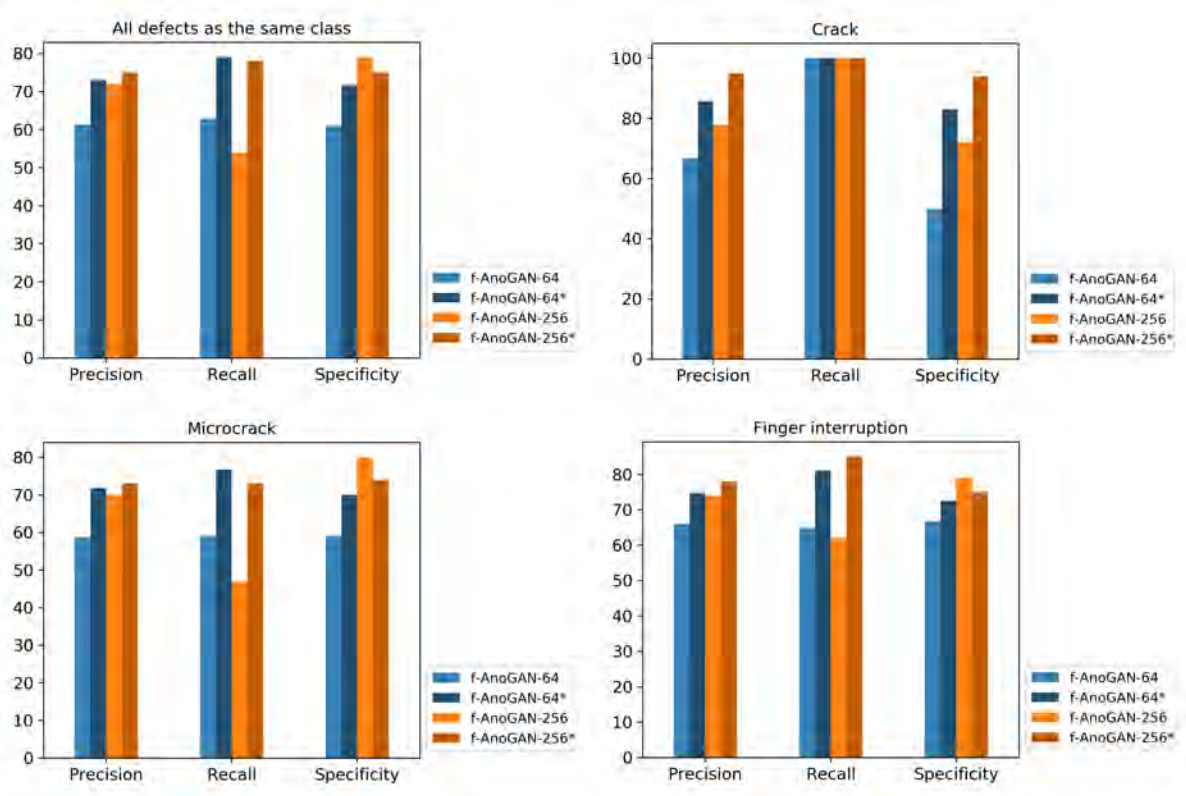


Figure 5.7: Detection rates results from each of the network configurations.

as all metrics improved. Nonetheless, the change is more noticeable for f-AnoGAN-64 than for f-AnoGAN-256 where the initial detection rates were higher. The detection rates from f-AnoGAN-64* were a bit higher than the ones from f-AnoGAN-256*, but it still required more than a second to process a cell, making it still not applicable for inspection. In the case of f-AnoGAN-256*, the architecture was not modified so the processing time was not altered leveraging the benefits of detection rates but remaining applicable under real inspection time requirements.

5.2 Automatic labeling

In anomaly detection, the model is taught to find everything that is not considered normal. In a supervised training like the ones presented in Chapter 4 the model is instead trained with labels to search for specific defective patterns in the data, which usually yields more precise models. Using the anomaly detection approach as an automatic labeling method, one may benefit from the precision of supervised learning models while

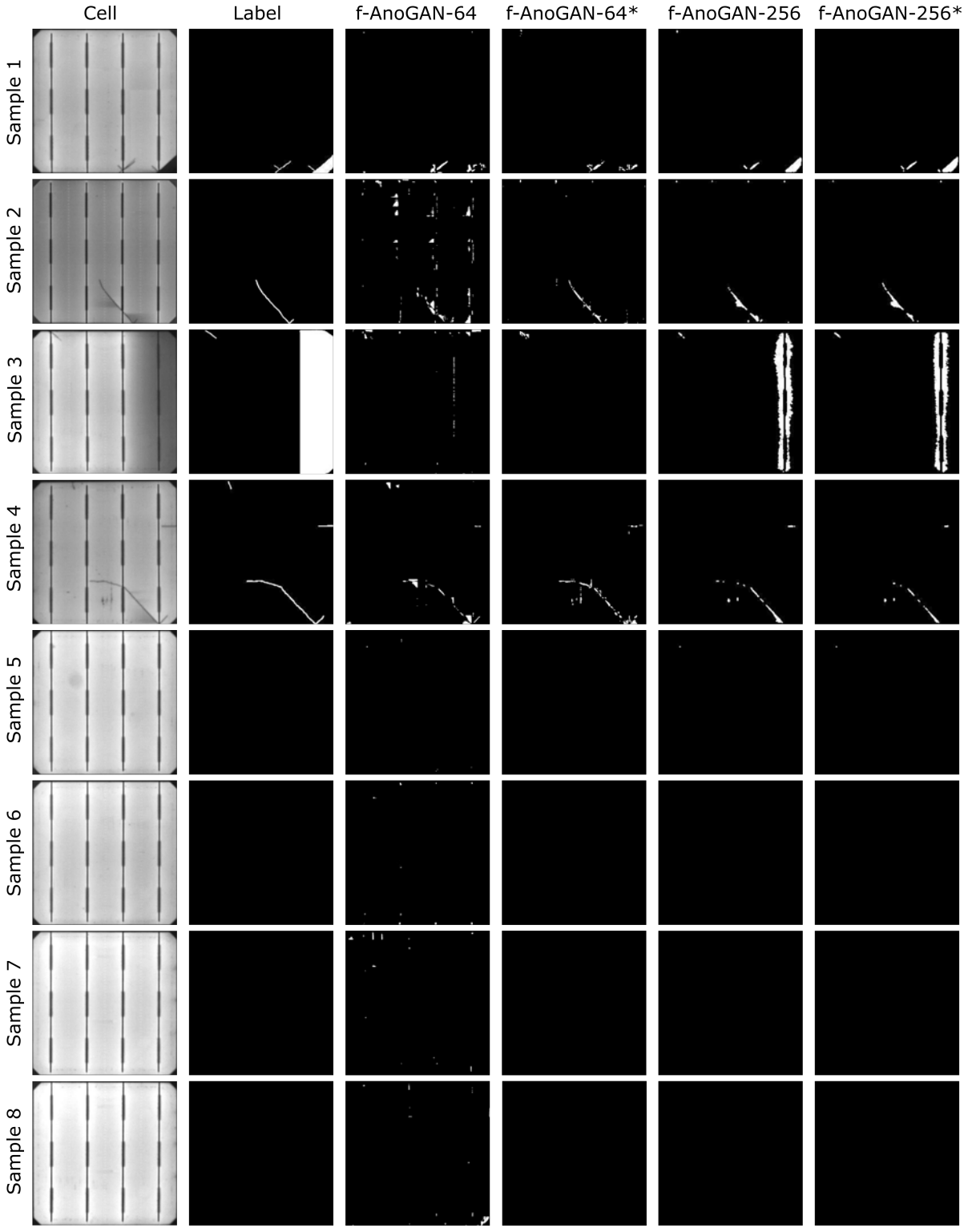


Figure 5.8: Pixel level defect detection of additional experiments.

avoiding the time-consuming, and not always trivial, pixel-level labeling task and thereby considerably reducing the effort dedicated to the setup of a new inspection system.

5.2.1 Experiments and results

The hypothesis of using the anomaly model as an annotator was validated in the following experiment where a network trained using expert generated manual labels and the mentioned automatic labels were compared. The automatic labels were generated with the model f-AnoGAN-256. Among the trained models, f-AnoGAN-256 showed the highest detection rates, short enough processing time for industrial inspection, and precise pixel-level results. Taking into account that the defect location results in Figure 5.6 were close enough to what human experts annotated, it was decided to choose this model as the automatic annotator. Some samples of the automatic labeling used for training are shown side by side with their corresponding manually labeled in Figure 5.9.

For both manual and automatic labeling models, the same U-net employed in Section 4.2 was used. The network configuration was kept as in the previous experiment. The networks were trained also to minimize the dice loss in Equation (4.2) and using Adam as the optimization algorithm with a learning rate of $1e^{-4}$.

Unlike the experiments with the anomaly model section, in the experiments in the supervised part, just defective samples were used for training. So the defective samples were split into train, validation, and test sets following the next distribution: 300 for training and validation, and the remaining 75 for testing (4 crack images, 48 microcrack images, and 23 finger interruptions images). In addition, the defect-free samples used in the evaluation in the unsupervised part experiments were also employed to evaluate the models in this section.

After training U-net separately with the two versions of the dataset (manual and automatic), the 75 defective and 375 defect-free samples were employed to compute the metrics and evaluate the performance of the models. The results are shown in Table 5.4. In addition to the U-net-based models, the model from the previous section (i.e., f-AnoGAN-256) was also executed on the same test to validate that the supervised training with automatic labels improved the detection rate compared with the anomaly model.

As shown in Table 5.4, both supervised models yielded higher detection rates than the anomaly detection models without incurring more False Positive cases. If supervised

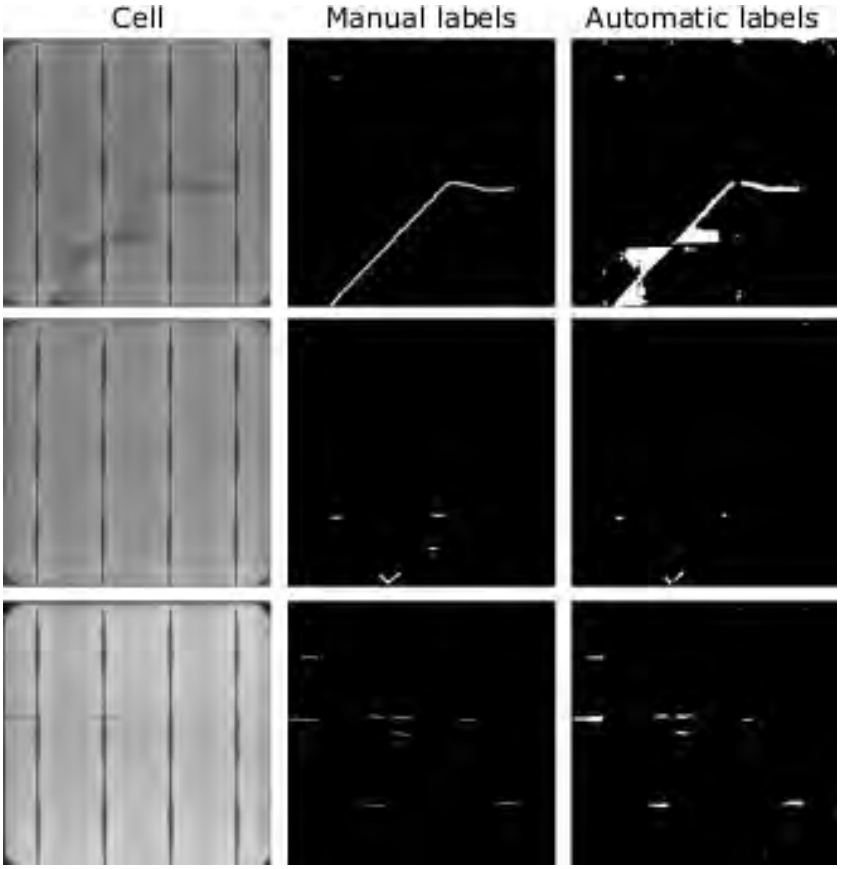


Figure 5.9: Manual and automatic labeling for different samples. The automatic labeling kept the segmentation of the labeled defects, but at the same time introduced some additional areas. This is especially noticeable in the samples from the first row, where the manual labeling only considered the defect itself, but the automatic labeling also considered the darker areas created by the effect of the defect.

Model	Recall	Precision	Specificity
U-net w/ manual labels	80	95	99
U-net w/ auto. labels	93	81	95
f-AnoGAN-256	79	73	73

Table 5.4: Image-level results from U-net trained on manually created labels, U-net trained on automatically created labels, and also, the results from the anomaly model used for annotation.

models are compared with each other, U-net trained with automatic labels was able to detect more defective samples (Recall of 93%) than U-net trained on manual labels (Recall of 80%).

However, using automatic labels resulted in more False Positive cases, making the Precision and Specificity values decrease from 95 to 81 and 99 to 95, respectively. Note

that the increase of False Positives has a larger impact on the Precision because of the imbalance of defective and defect-free samples in the test set.

As for the segmentation results, some samples are illustrated in Figure 5.10 where Samples 1, 2, 3, and 4 are defective samples with defects contemplated at training and metrics evaluation, and samples 5 and 6 are defect-free samples. Samples 7 and 8, instead contain defects that were not considered during training and testing but illustrate the effect of the automatic labels in the segmentation results.

As can be seen that the defects were more thoroughly marked with U-net trained with automatic labels than with manual labels. The second and third samples in Figure 5.10 are an example of this. However, impurities in the cells that were not taken into account during manual labeling were also detected as defects (e.g., black spots under the crack in the second sample or around the finger in the third sample). This caused certain defect-free samples with such impurities to be classified as defective cells, which increased the number of False Positives resulting in an impact on the image-level metrics. Nevertheless, few defect-free samples present these False Positive cases.

In addition, even when not considered during training and when the metrics were calculated, the automatic labels enable U-net to segment other kinds of defects in the seventh sample where both models were able to detect the microcrack, but the break at the bottom right was only detected by the models trained with automatic labels. The same happened in the eighth row sample where the bad soldering was not segmented when using manual labels.

Concerning the annotations, it seems that annotating dark areas around the defects has a positive effect on the models' pixel-level results. For example, in the first sample in Figure 5.9, the manual label does not cover the areas around the defect, whereas, with automatic labeling, these areas are annotated as defective. The experts did not consider these areas during the labeling as they are not part of the defect, but a consequence of the defect itself. However, these dark areas will not appear in defect-free cells. Because of that, the anomaly detection network annotated them as defective areas. When considering these dark areas as part of the labels, the network trained on automatic labels recognized dark areas around defects as defective.

Consequently, as shown in the eighth sample in Figure 5.10, even if the class was not included in the training, the dark area on the right that belongs to a bad soldering defect was segmented when using automatic labels and not when using manual annotations. The same happened with the break in the first sample. Moreover, the segmentation of

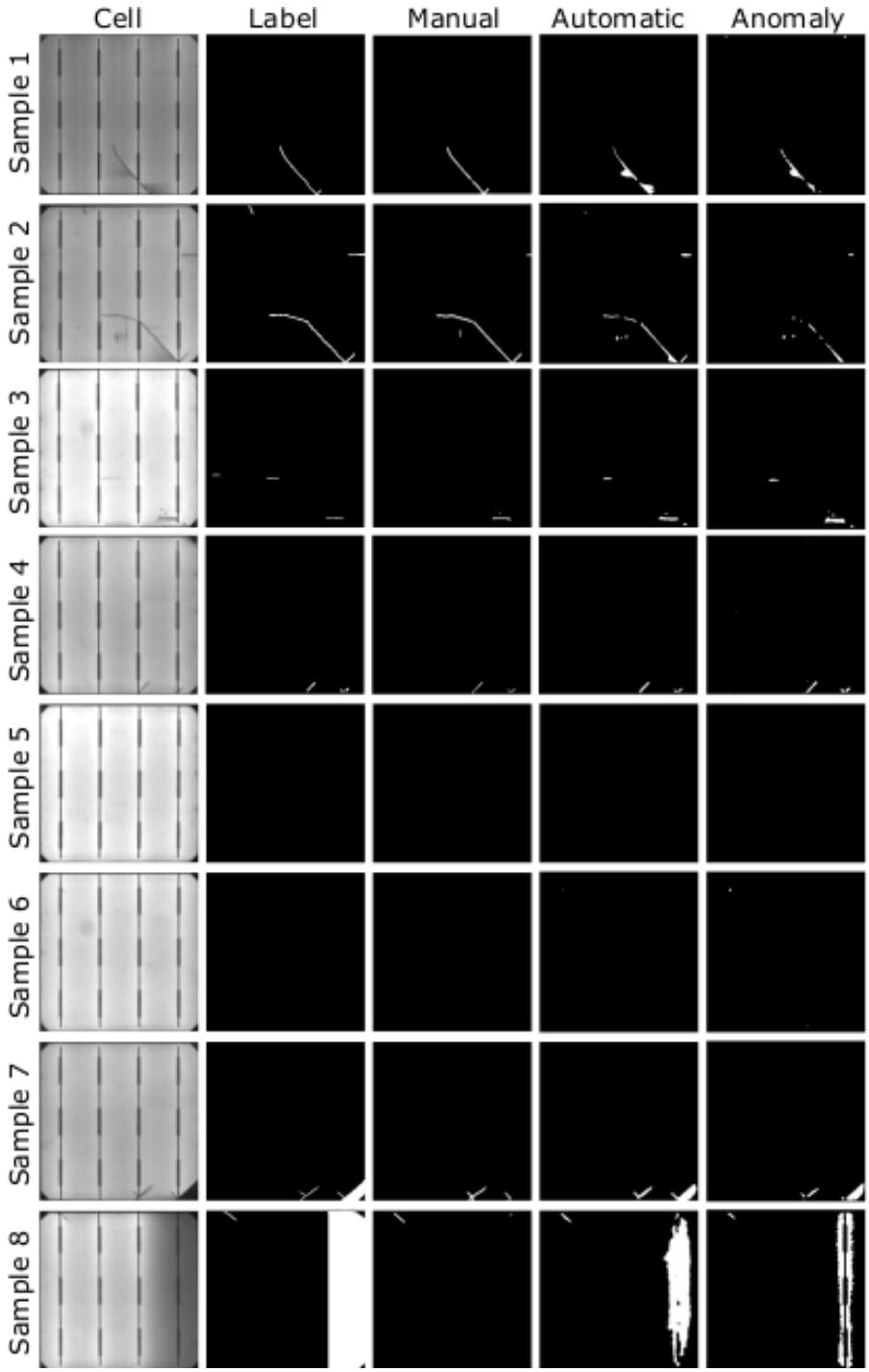


Figure 5.10: Results from supervised training models and from the anomaly model used for annotation for comparison. Label refers to the annotation made by experts, manual refers to the segmentation results obtained from the supervised segmentation model trained with manually labeled samples, and automatic refers to the segmentation results obtained from the supervised segmentation model trained with automatically labeled samples as ground truth.

other defects, like the finger interruption in the third sample and the microcrack in the fourth, have been more accurately segmented. Nonetheless, by annotating dark areas as defective, certain impurities that were not considered as defects were also segmented. So, including dark areas as part of the labels was beneficial for pixel-level results and to detect more defective samples, even if it made some new False Positive cases arise.

5.3 Concluding remarks

In this chapter, it has been shown that the anomaly detection approach provides practitioners with a tool to obtain an inspection model using only defect-free samples as training data. This feature is key for the development of a PV module inspection system as it permits companies to have an inspection model from the very beginning stage of a new production line setup, without waiting for defective data to appear.

In order to apply anomaly detection for industrial inspection, a GAN proposed to detect and locate anomalies in the medical domain has been adapted. The adaptations have been two-fold: First, the architecture has been modified such that the images can be processed in a single step instead of processing them by patches. In this way, less time is required to process a cell, and therefore the established inspection time mark of less than a second per cell has been met. And second, the training scheme has also been modified. This modification has resulted in an improvement in the defect detection capabilities of the model.

Furthermore, it has been experimentally demonstrated that the results from the anomaly detection are potential pixel-wise labels that can be used in a supervised training. In the experiment, the defect localization results obtained from a model trained with expert generated labels and a model trained with automatically generated labels have been compared. The comparison has shown that using automatic labels is comparable to using manual annotations, thus, it is feasible to use anomaly detection as an automatic annotator which can notoriously reduce the effort needed to prepare an inspection system.

Chapter 6

Model Adaptation

The previous two chapters have described the application of DL for anomaly detection purposes using unsupervised learning, and for classification and segmentation using supervised learning. Also, it has been described how the anomaly detection approach can be used to obtain automatic annotations for a supervised learning which avoids the need for manual labeling.

With both models, the anomaly model and supervised model, practitioners can build a robust quality inspection system that can detect defects in the production line, and at the same time, check for new anomalies in the cells. This setup would work flawlessly if the production lines will remain unaltered. However, industrial environments can be very dynamic and present situations where models need to be adapted to some changes.

For example, a customer might come asking to introduce some variations in the produced cells, like cells made of a different material or that contain a different internal structure. In this case, the trained model could experience some difficulties handling new case particularities by evaluating them always as anomalies or maybe as false positives in the case of the supervised model.

Another new situation could consist in the need for incorporating a new defective pattern to be detected by the supervised model, but not having access to enough samples of that particular pattern to retrain the model. For example, while the supervised model is detecting defects that were part of the training dataset, the anomaly model could

detect a new specific anomaly pattern on a recurring basis. After some analysis, this new anomaly type could become relevant and its detection necessary.

For the first situation, one possible solution could consist in repeating the unsupervised training followed by the supervised training described in the previous chapters to obtain the anomaly model and supervised model for the new domain. In the second case instead, a possible solution could be to wait until the anomaly model detects enough instances of the new anomaly, and at that moment, retrain the supervised model with the new class as part of the training set. However, these solutions might take some time in order to adapt everything, which is not always desirable.

The following sections will focus on describing two techniques as alternative solutions for these cases that will not require much time in their application. On the one hand, how to take advantage of already trained models as a starting point and thus achieve inspection models for the new domain in a fast way will be described. On the other hand, a technique with which it is possible to incorporate new classes into a model using a few defective samples for that will be described. These techniques will compose the final stage of the proposed methodology which consists in adapting the models obtained in the previous stages to changes in production.

6.1 Transfer Learning

When training a neural network, the main objective consists in making the network learn to extract meaningful representations from the data such that it can successfully perform the given task. The ideal scenario for such training would contain a big and diverse training dataset for the network to be able to learn. However, in a real world scenario, it is not always easy to gather enough data samples for every possible defect or even enough defective samples limiting in this way such training. In order to overcome such limitations and still be able to train a neural network when there is a scarcity of training data samples, there is a technique called Transfer Learning (TL) that takes advantage of the already acquired knowledge from a trained network as a way of network initialization.

[113] describes how after completing the training, the first layers in the network tend to focus on extracting simple and general features (e.g., edges, borders...etc) that are shared across domains, and as you go deeper, the layers start focusing on extracting more domain specific and complex features (e.g., eyes, noses...etc). TL leverages the general feature extraction capability from already trained networks by transferring the weights

into non-trained networks as a form of network initialization. In this way, the network can in some way avoid spending the initial training iterations and focus on learning to extract domain specific features, which will overall accelerate the optimization process.

In [29] they define two ways of performing the Transfer Learning: 1) training the new classifier (last layers in the network) while keeping transferred layers frozen [119, 74], or 2) performing a fine-tuning of the transferred layers using a low learning rate [2, 20, 109]. The first approach consists in employing the transferred layers as a feature extraction module and just replacing the trained classifier with a brand new classifier that will be randomly initialized. Then, the training will only have an effect on the weights in the added new classifier as the other layers will be kept unaltered. The second approach instead, consists in using the weights from the trained network as a kind of specialized initialization such that just a few updates will be required to adapt the network to the target domain. In order to avoid "overwriting" the transferred learned pattern extraction capabilities, a very low learning rate is usually employed so the weights only experiment with small variations. Figure 6.1 shows a simple schema of these two types of Transfer Learning.

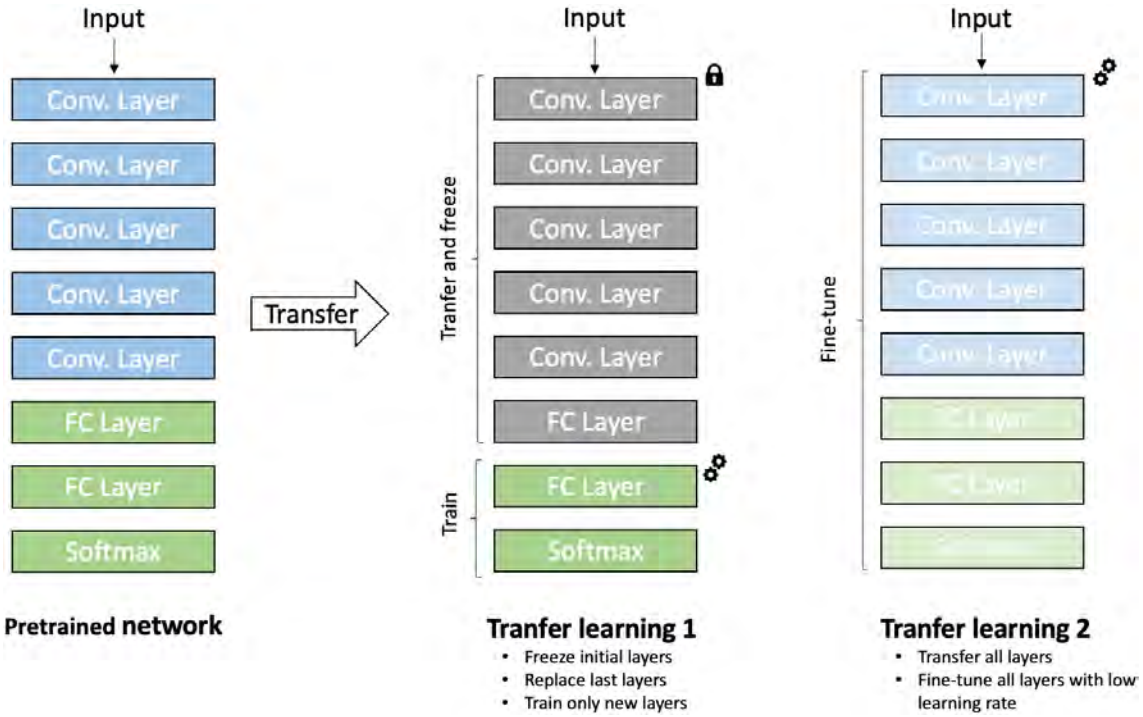


Figure 6.1: Different Transfer Learning schemas.

Moreover, both forms of TL can be sequentially applied. First, transfer the initial layers, freeze them, and only train the added new classifier, and then, once the classifier is

trained, unfreeze the initial layers and slightly tweak all the layers with a very low rate to end adjusting the network to the target domain. It is recommended not to mix both Transfer Learning schemas at the same time as a randomly initialized classifier can lead to high error values that will be backpropagated through the network making the network lose the transferred pattern extraction capabilities.

As stated, both Transfer Learning approaches reduce the need for training data, nevertheless, the fine-tuning approach is a more suitable solution for scenarios where more training data is available as all layers in the network will be part of the training process and this will require more data. Instead, in the other approach, just the parameters in the new classifier will be modified, which will demand a lower amount of data samples. Based on this, the freezing approach was used for the experiments in this section.

6.1.1 Experiments and results

The experiments in this section have consisted in using different dataset combinations to explore the applicability of TL in our datasets. Three different experiments have been conducted in order to test the use of transfer learning methods in this thesis context: transfer from monocrystalline cells to polycrystalline cells, transfer from polycrystalline cells to monocrystalline, and a transfer from a model trained on polycrystalline cells with 3 and 4 buses to cells with 5 buses.

The three experiments wanted to show how a model trained on a source domain can be adapted to work on the target domain using just a few defective data from the target domain in the adaptation. The experiment was done as follows: first, a base model was trained using all the defective data available in the source domain. For instance, in the first experiment where the transfer was from the monocrystalline cells to polycrystalline cells, the base model was trained on all the data available from the monocrystalline cells dataset. Then, once the base model was trained, few samples, 20 in this case, were selected from the target dataset and the transfer was performed. As stated above, the transfer was done using the layer freezing strategy. Finally, the adapted model was executed on a test set composed of the samples from the target domain to extract quantitative metrics as well as some segmentation results.

In all the experiments the transfer operation using the supervised sliding window based and the U-net models from the supervised training chapter 4 following the layer freezing strategy is performed. In the case of the sliding window based approach, the convolutional

blocks in the network were frozen, and in the case of U-net, the layers that composed the encoder part were frozen. Moreover, the objective of the experiments consisted in comparing the metrics obtained for models adapted to the target domain through Transfer learning against the metrics from models in supervised training chapter 4 that were directly trained on the target domain. Taking into account this, the samples for the transfer operation were carefully chosen thus the same test set used in the supervised chapter 4 experiments could be replicated. Note that in the case of U-net based models, just defective samples were used both for transfer and training the base model as it showed that this model had troubles when employing defect-free samples for training.

In the next subsections, the results obtained in each of the experiments performed are going to be described.

Transfer from Mono to Poly

This first experiment consisted in training the base network on the monocrystalline dataset in ordinary supervised training and then using it to perform the transfer on the polycrystalline cells. As the samples from the monocrystalline were not required at testing, all the available samples were used to train the base network. The quantitative results from these experiments are shown in Table 6.1.

	Recall	**Precision**	**Specificity**
Poly. base slid. Wind.	100	69.5	73
Mono. to Poly. slid. Wind.	83	72	78.8
Poly. base U-net	87.5	82.3	88.4
Mono. to Poly. U-net	86	89	93

Table 6.1: Results from training both approaches, first if the network was trained directly on the target dataset (i.e. polycrystalline cell images), and then, the same network but using Transfer Learning (i.e. from monocrystalline cells to polycrystalline cells). Note that the base results are taken from the supervised chapter 4).

As can be observed, the transfer from monocrystalline cells to polycrystalline cells yielded similar results to the ones obtained through "ordinary" supervised learning. The network with Transfer Learning was able to detect fewer defective samples (lower Recall values) than the completely supervised model by just a small margin. This is more appreciable in the case of the sliding window based networks where the value of Recall dropped

from 100% to 83%. The base sliding window based network was able to detect all defective samples but this was at expense of a greater number of False positive cases (lower Precision and Specificity) than the same network with the transferred weights. In other words, as more areas were classified as defective more defective samples were finally detected correctly, but at the same time, more areas in defect-free samples were classified as defective i.e., FP cases. With regards to the U-net network, a similar effect can be appreciated. The base network detected more defective cases, but it incurred in more False positives cases, whereas in U-net with transferred weights less defective samples were detected but also a lower number of False positives were obtained. With respect to the qualitative results, Figure 6.2 shows a couple of examples of defective and defect-free cell segmentation results.

Figure 6.2 shows how the networks had large FP cases, especially the sliding based approach that confused all the buses as they were defects. Nonetheless, these results were expected due to the difference between the monocrystalline and polycrystalline cells. If a network that has been trained on monocrystalline cells is employed directly on polycrystalline, it can be anticipated that background noise (i.e., crystal grains) will confuse the network and yield a large number of FP cases. Nevertheless, after the Transfer Learning, almost all these False Positive cases were almost completely removed from the predictions. There were certain cases, some areas from the sliding window based results that were still classified as defective (e.g., fourth sample). However, these cases have no major relevance in overall results as they can be easily removed using a proper threshold taking into account that they present lower accumulated probability values that the prediction over the real defects.

Transfer from Poly to Mono

In this second experiment, the based models were networks trained with polycrystalline cells and transfer consisted in adapting them to monocrystalline cells. The quantitative results from this experiment are presented in Table 6.2.

The results in the table show that there is not much difference between training directly on the target domain (i.e., monocrystalline cells) or performing a transfer from polycrystalline cells. The results were not much different in either of the networks, maybe a slight improvement can be appreciated in the case of sliding window based network results, and a slight increase of Recall at expense of a decrease in Precision in the case of U-net network.

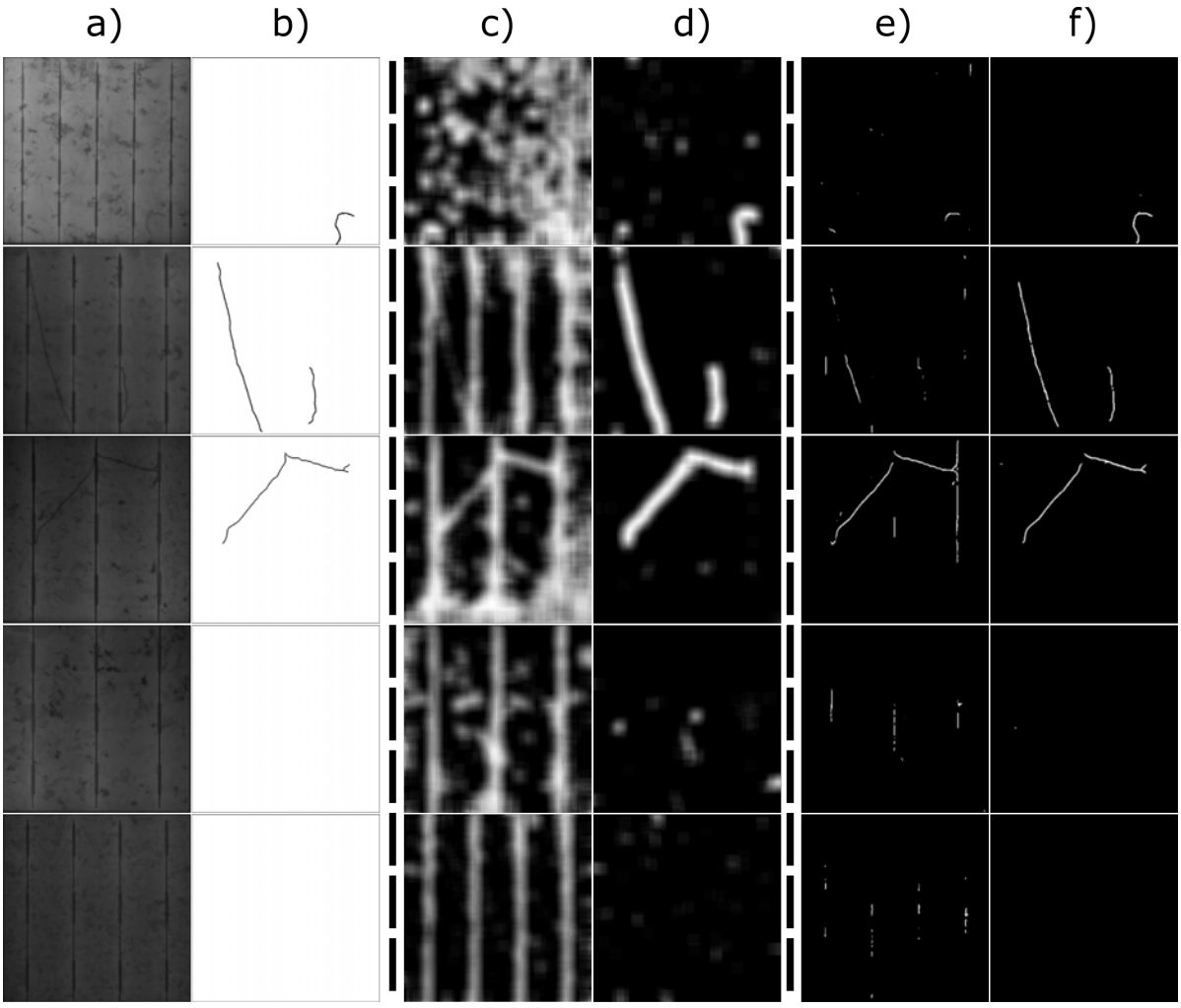


Figure 6.2: The results of the networks before the transfer (trained just on mono. samples) and after the transfer. From left to right: a) the EL image of the cell, b) the pixel level manual label, c) the result of sliding window based network on the poly. cells before the transfer, d) the result from the same network after the transfer is performed, e) the result of U-net on the poly. cells before transfer, and f) the result from U-net after the transfer is performed.

Regarding the segmentation results, some examples of defective and defect-free samples are presented in Figure 6.3.

The results in this experiment were less different than the ones obtained in the previous section. Even though the networks were not specifically trained on monocrystalline cells, they did not have as much FP as when the networks were trained on monocrystalline and tested on polycrystalline. However, as with the quantitative results, these results were more or less expected. If one takes into account the fact that the base network was trained on polycrystalline cells where there are both defects and crystal grains in

	Recall	Precision	Specificity
Mono. base slid. Wind.	86	74	94
Poly. to Mono. slid. Wind.	87	77.6	95
Mono. base U-net	80	95	99
Poly. to Mono. U-net	81	89	98

Table 6.2: Results from training both approaches, first if the network was trained directly on the target dataset (i.e. monocrystalline cell images), and then, the same network but using Transfer Learning (i.e. from polycrystalline cells to monocrystalline cells). The base results are taken from the anomaly detection Chapter 5

the cells, and also that the network must have learned to pay attention only to defects. Correct segmentation results and FP cases close to zero can be expected to be obtained as if the same network were run on monocrystalline cells that have no structures but defects.

The only place that showed to be problematic for the networks was the corners of the cells. Polycrystalline cells do not have corners like monocrystalline cells. Because of this difference, before the adaptation, the networks considered these structures as they were part of a defect since they present a big gradient similar to the one that can be found in real defects. This is more noticeable in the case of the sliding window based network results. Nonetheless, this effect disappeared after the transfer was performed for both cases. Also, it seemed that the transfer improved the segmentation for certain defects such as the microcrack at the bottom-left of the third cell.

Transfer from Poly with 3 and 4 buses to 5 buses

The last experiment focused on a subtle case of transfer. In this case, the objective was to adapt a model trained on cells from the polycrystalline dataset with 3 to 4 buses to cells with 5 buses. As mentioned in Section 2.3, nowadays the solar industry is moving towards an increase in the number of buses in the cells as it has been proved that in this way the performance that the panel shows at generating energy improves. This experiment simulates a situation where a company that has been producing panels with cells of 3 and 4 buses might want to extend the production to also generate panels with cells with 5 buses in order to provide the clients with a more advanced class of panels.

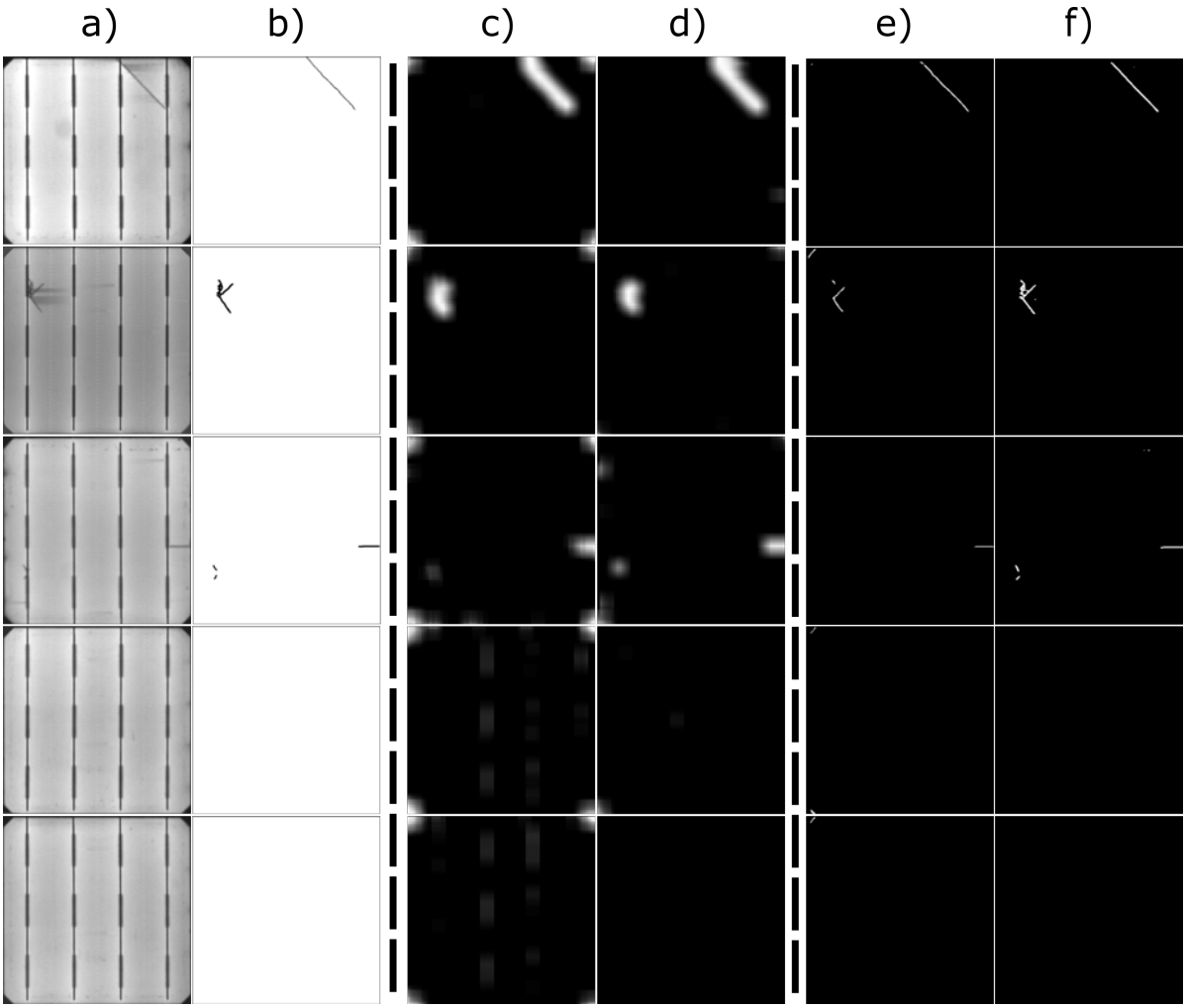


Figure 6.3: The results of the networks before the transfer (just trained on mono. samples) and after the transfer is performed. From left to right: a) the EL image of the cell, b) the pixel level manual label, c) the result of sliding window based network on the mono. cell before the transfer, d) the result from the same network after the transfer is performed, e) the result of U-net on the mono. cell before transfer, and f) the result from the same network after the transfer is performed.

Unlike in the previous experiments, in this case, there were just 7 defective samples available with 5 buses. Thus, the transfer was performed using only four samples of the cells with 5 buses, and leaving in this way at least 3 samples for testing. It was considered that 3 samples are not enough to compute any sort of representative quantitative evaluation, so it was decided to just report qualitative results exhibited in Figure 6.4.

As it was anticipated, when executing the network before the transfer the buses were considered as they were a kind of defective structure. It is interesting to see that the

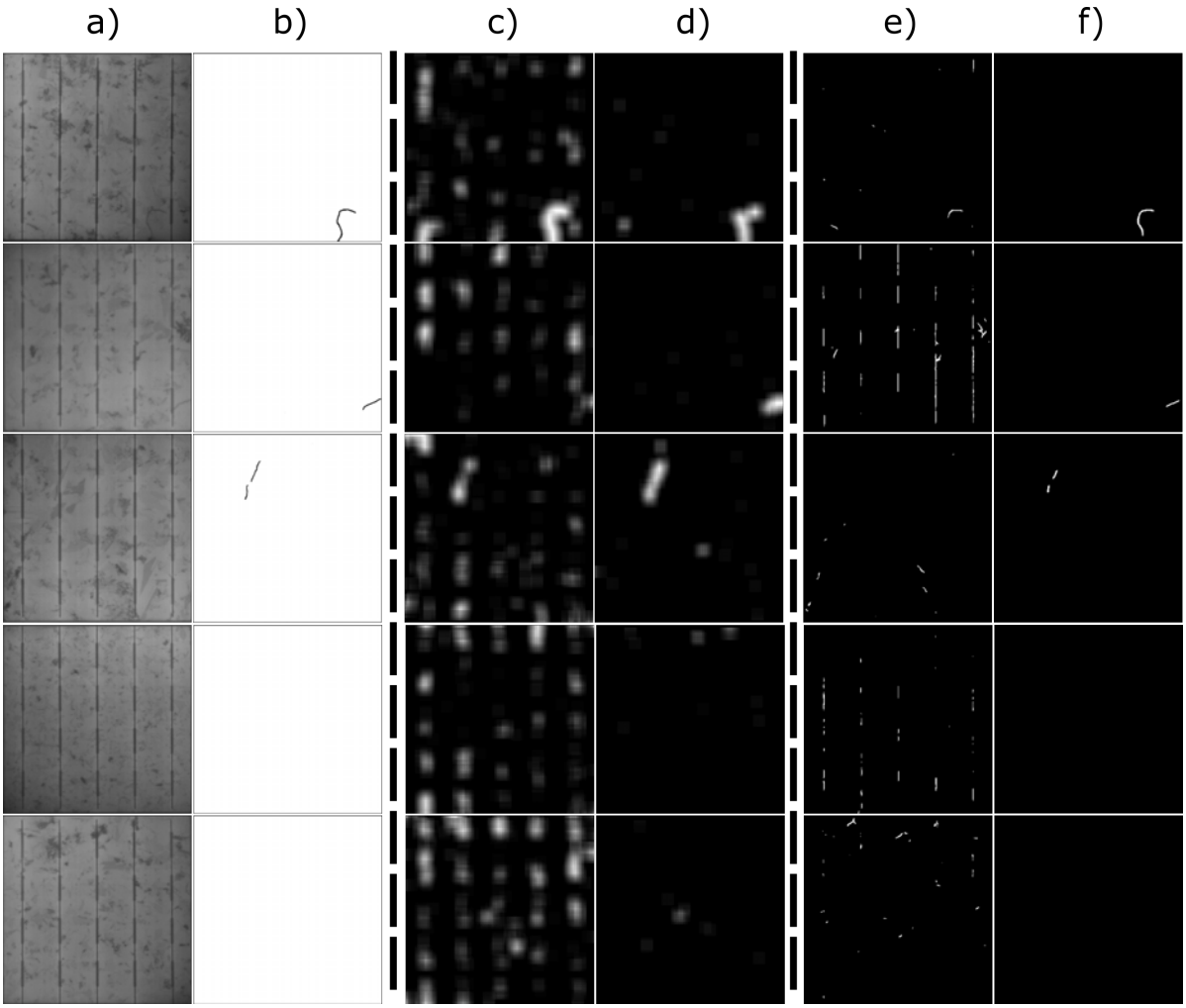


Figure 6.4: The results of the networks before the transfer (just trained on poly. samples with 3 and 4 buses) and after the transfer is performed on cells with 5 buses. From left to right: a) the EL image of the cell, b) the pixel level manual label, c) the results of sliding window approach based network on the poly. cell with 5 buses before the transfer, d) the results from the same network after the transfer is performed, e) the result of U-net on the poly. cell with 5 buses before transfer, and f) the results from U-net after the transfer is performed.

sliding window based network also performed badly on the buses. It was thought that when using the sliding window, the network could be abstracted from the global structure and not differentiate a patch coming from either of the cells with 3, 4, or 5 buses. In other words, by processing the cells by patches, it should not be affected and perform almost identically in any sort of cells if they were from the same family (i.e., polycrystalline or monocrystalline). However, the results show that the whole structure has an effect on the network performance. The differences in bus structure between cells may not be visible to the naked eye. But at the pixel level, there may be some variations, either at

the structure level or at other levels (e.g. illumination) that can affect how the model acts. In the case of U-net based network instead, the performance was as expected due to the way the network processes the images. When processing the whole image in one forward pass, the number of buses in the cell completely changes the global structure and as a consequence affects how the features are extracted. Nevertheless, after the transfer, the False positives were removed from the results leaving just defective areas in the predictions.

Transfer Learning is an interesting method to obtain a model where there are not lots of samples available for ordinary supervised training. The results in this section have shown that this method is actually applicable in the solar cell industrial domain in a variety of data configurations, which means that it really can save resources to the companies. Nonetheless, it should be taken into account that Transfer Learning might still require a certain amount of defective samples to be applicable.

6.2 Few-shot learning

In recent years, a new method denominated Few-shot learning has been gaining the attention of the DL community as it focuses on making neural networks learn to deal with new classes at testing time in a data scarce regime. Although this section will be focused on Few-shot learning, in the literature this method can also be found under the tags of One-shot or Zero-shot learning which in the end seek the same objective but under more strict data availability.

Few-shot learning has been employed for different well known computer vision problems like image classification, object detection, or image segmentation. However, as this thesis has focused on the image segmentation task, just proposals that were designed for this specific problem will be considered.

The first to apply this method was [85], where they proposed a two-branch FCN (base branch and auxiliary branch) architecture. While the base branch, an already trained network, is processing a query image (an image that contains the new class), the auxiliary branch processes a support set that will serve to extract guidance parameters that will guide the base network in segmenting the query image. The support set in this case would be composed of a set of pairs of images and pixel-level annotations, that will contain other instances of the new class that will be trying to segment. In other words, if our new class consists of a "cow", the query image will contain a cow for example in a

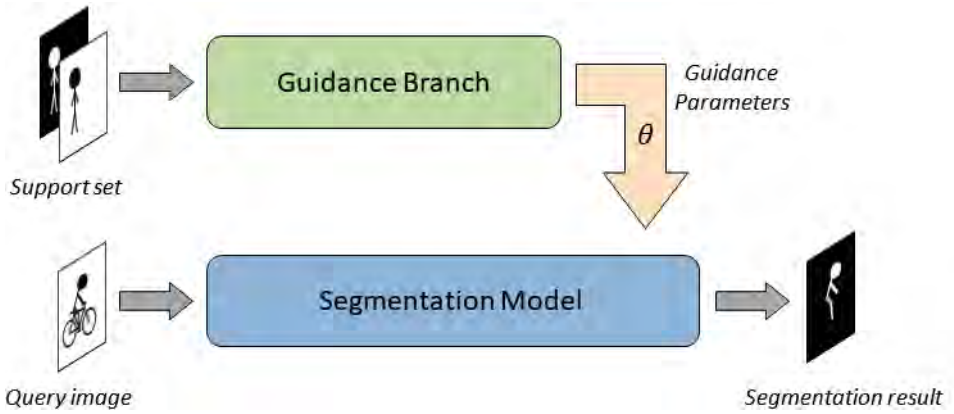


Figure 6.5: Schema of two-branch based network for few-shot image segmentation.

meadow, and the support set will contain pairs of images and pixel level annotations of cows in other locations or positions within the image such as cows in a farm, in the road...etc. Figure 6.5 illustrates a simple sketch of how this method works.

From that paper on, several works have been published [79, 111, 35, 115, 117] where they also follow this two-branch based network architecture. Nonetheless, in these works, they do not contemplate a continuous learning scenario where the network would be increasingly incorporating the capabilities to segment new classes. The learning process is limited to guiding the network in segmenting the instances from the new class without paying attention if the network is forgetting how to segment previously learned classes. From an industrial application perspective, a continual adaptation of the system would be a more interesting approach since both, old and new class instances, will be important to be detected in production. Thus, this section has focused on the incremental learning scenario where the network needs to incorporate the capability of segmenting new classes but it still needs to be able to keep segmenting old class instances.

Specifically, the experiments have focused on the recent proposal in [86] as it contemplates both Few-shot learning and incremental learning scenarios. For this section, the proposal at [86] which employs the "weight imprinting" technique as a way of incorporating new classes into the network without any additional training procedure has been explored. All the experiments have been carried out in the industrial context of solar cell quality inspection. The experiments have consisted in training a network on the previously used defect classes (i.e., cracks, microcracks, and finger interruptions), and trying to extend its capabilities to also be able to work with defect classes that have only a couple of samples in our data set (i.e., black spots and bad soldering). In addition, the original network in [86] has also been modified with the goal of improving segmentation results.

The work in [86] is deeply rooted in two papers: Proxy-NCA in [72] and the proxies method applied to few-shot classification in [75]. In the first paper, they proposed a proxy-NCA loss function as a way of reducing the computation burden, thus, speeding the time to converge of methods applied in metric learning that use *triplets* based loss. Triplets loss aims to minimize the distance between similar points and maximize distance with dissimilar points using pairs of triplet points (anchor point, a positive similar point, and a negative dissimilar point). If there are a lot of points in the training set, the combinations of triplets that can be composed can rapidly escalate making the optimization problem hard to be solved. In order to reduce the amount of possible triplets combinations, they proposed to use sets of *proxies* as the representative point for every class. Thus, instead of computing the loss for all possible triplet combinations, the triplets combinations will be reduced to each point, the positive proxy of the point, and the different negative proxies.

Following this idea, in [75] they established a similarity between the proxy-NCA loss and the Softmax cross-entropy loss used in neural networks if both point vectors and proxy vectors are normalized to the same length. They argued that, if both vectors are normalized to the same length, minimizing the euclidean distance between a point and its proxy is equivalent to maximizing their cosine similarity. Thus, the euclidean distance in proxy-NCA loss can be substituted by the cosine similarity, making the loss resemble the Softmax cross-entropy. Following this similarity, they proved that normalized embeddings from the neural network can act as weights for a new class in the final fully connected layer of the network, such that one example will be sufficient to extract embeddings and perform what they called *weight imprinting* to the number of classes in the classifier.

However, unlike the classification, in the context of segmentation the embeddings have not a vector shape but three dimensional shape containing features from different classes. To adapt the *weight imprinting* from classification to segmentation, [86] proposed an architecture that incorporated a Normalized Masked Average Pooling (NMAP) layer where the output embeddings are pixel-wise masked for the new class with a given support set that contains samples with instances of the new class and their corresponding labels and then are averaged and normalized by procedures in Equation 6.1 and Equation 6.2. In this way, just relevant features for the new class remain in embeddings that will be used as the proxy for the new class c.

$$P_c = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{N}\sum_{x\in X} F^i(x)Y_c^i(x), \tag{6.1}$$

$$\tilde{P}_c = \frac{P_c}{\|P_c\|_2}, \tag{6.2}$$

where Y_c^i is a binary mask for the i^{th} image with the new class c in the support set, F^i is the feature maps for i^{th} image. X is the set of all possible spatial locations and N is the number of pixels that are labeled as foreground for class c. Moreover, they extract features at different layers in the network when computing the proxies so information at several resolutions is taken into account, and thus improve the final segmentation result. The network architecture is shown in Figure 6.6. The base architecture is VGG-16 [87], with additional skip connections to extract different resolution embeddings which are composed of 1x1 convolution layers. The embeddings at these layers are mapped to the label space to then perform the weight imprinting for the new class.

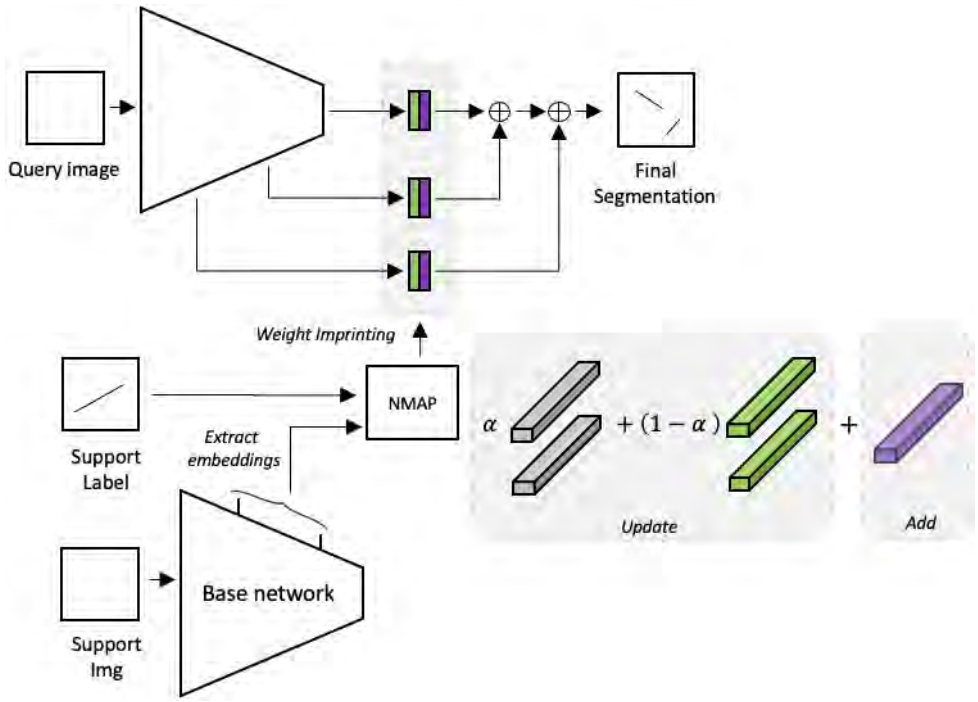


Figure 6.6: Original network architecture for few-shot segmentation in [86].

In addition, they also considered the continual learning scenario by updating weights from the learned classes with the information coming from the support set at every imprinting procedure. When an instance of a learned class is present in the support set along with the new classes, the embeddings from that instance are also extracted. Then, at imprinting, the weights from old classes are updated with their corresponding proxies by Equation 6.3.

$$\tilde{W}_c = \alpha \tilde{P}_c + (1 - \alpha) W_c \tag{6.3}$$

where P_c is the normalized proxy for the class c, W_c are the weights from the previously learned classed at the 1x1 convolution layers, and α is the updating rate. In this way, as new samples from the last imprinted classes arise, the features that were not present when the first imprinting was performed can be incorporated into the network. Thus, the weight related to the new class can be consolidated.

For this section, first, the architecture in [86] was employed on our dataset, and then the architecture was modified so it resembled U-Net shape as illustrated in Figure 6.7 aiming to improve the segmentation results. After some first tests, it was seen that the original architecture yielded poor and coarse segmentation results regarding the shape of big defects and the number of detected defects. Precise results are key for establishing the severity of the defect and determining if the cell needs to be completely discarded or could be repaired to put it back in the panel.

The original U-Net architecture was designed to follow an encoder-decoder shape with skip connections between the blocks in both parts. These connections were incorporated to allow the network to use the full potential of the features for the segmentation. In this experiment, the network was extended by adding more skip connections composed of 1x1 convolutional layers to perform the weight imprinting. Figure 6.7 shows this extension where the original connections are in gray and the new ones in green and purple.

6.2.1 Experiments and results

Before modifying the network architecture, first, it was experimented with the original network on our dataset to check if it was applicable and also to establish some base results that were needed to improve with the modification. The first experiment was carried out by training the network on the base classes in the training set, i.e., cracks, finger interruptions, and microcracks, and then, executing the trained network on the test samples that contained the same defects. The results of this first experiment are shown in Figure 6.8.

These first results show that the original network (i.e. FCN8) was able to locate almost all defects in the cells. In the case of the smaller defects such as the finger interruptions in the third sample in Figure 6.8, the FCN8 network could not segment any of them. In addition, even though bigger defects were detected, the network output a coarse

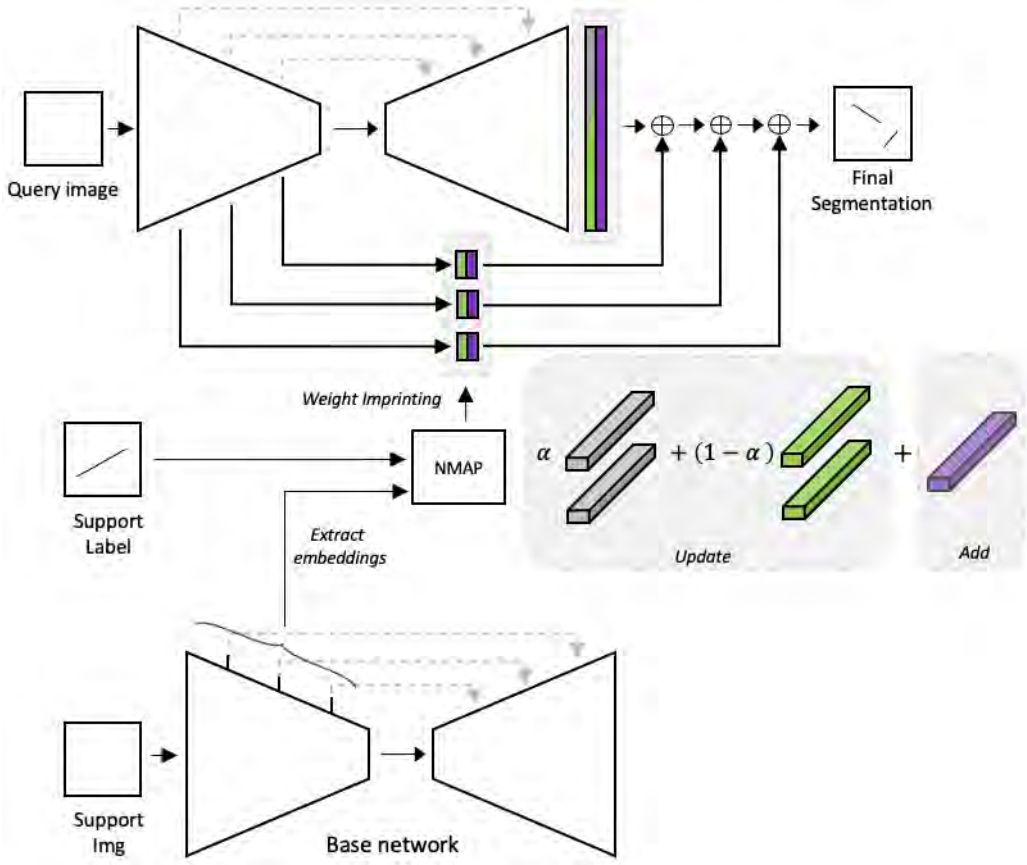


Figure 6.7: The adapted network for few-shot following U-net architecture.

segmentation. After these results, the architecture was substituted with a U-net based network, and the same training and testing were performed.

As mentioned above, U-net was extended with extra skip connections to allow us to perform the weight imprinting. The rest of the architecture with respect to the blocks at the encoder and decoder parts was kept as in [81]. The training was performed using the cross-entropy loss and RMSprop optimization function. In this case, weighted cross-entropy was required for training to alleviate the effect of the unbalance between defective and non-defective pixels, especially for the cases of microcrack and finger interruptions where the defective pixels suppose less than 0.1% of the pixels in the samples.

As can be seen in Figure 6.8, the segmentation results with U-net were more refined than with the original network, and also, the lack of detection with respect to the smaller defects was solved.

After the base training, two sequential imprinting operations were performed: first the black spots defect class was incorporated, and then, the bad soldering defect class was

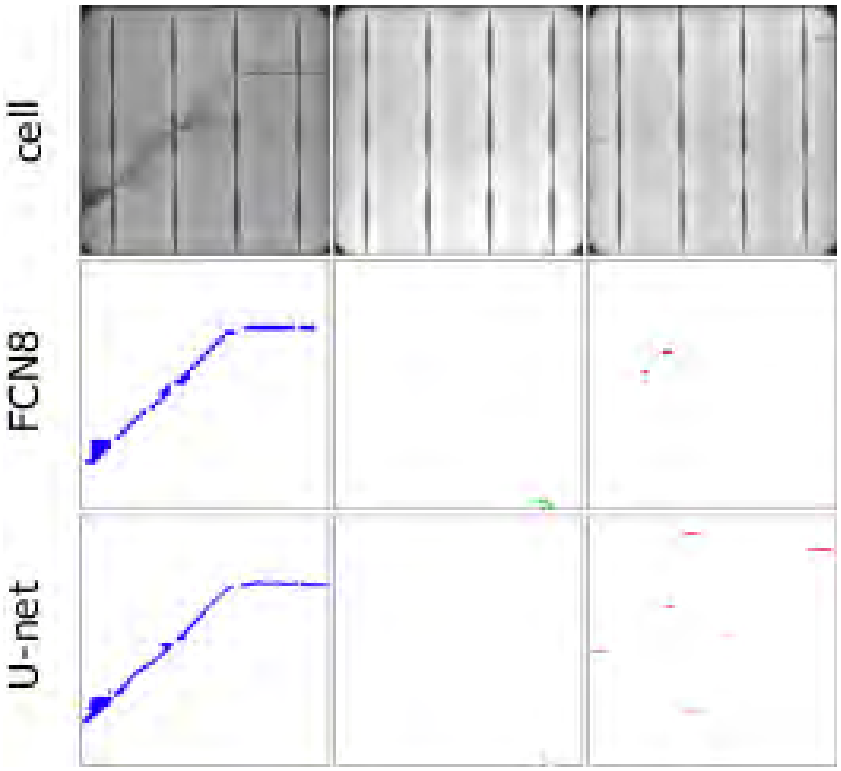


Figure 6.8: Results on the base defect classes (cracks, microcracks and finger interruptions) with the original network and with the U-net based architecture

incorporated. Figure 6.9 exhibits two examples that contained instances of these two defect classes. In the end, a network that can segment 5 classes (3 base + 2 new) was obtained. For this operation, the samples allocated for imprinting in Table 6.3 were employed.

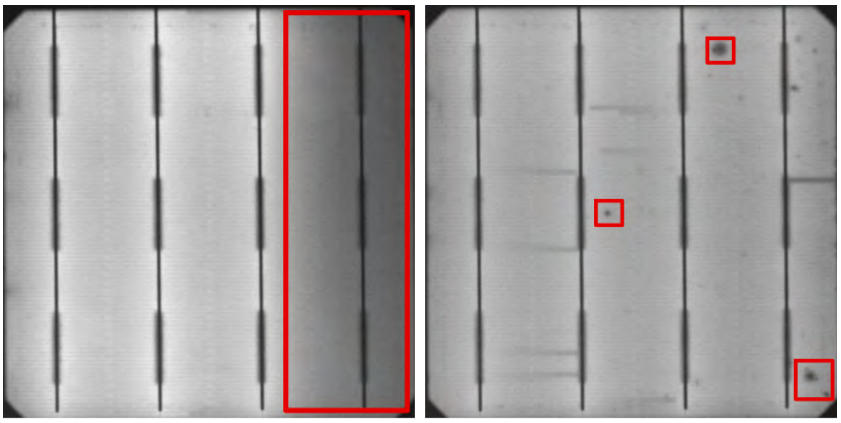


Figure 6.9: Bad-soldering (left) and black spots (right) defect classes examples highlighted with bounding boxes.

	Train	Imprin.	Test	Total
Defect-free	-	-	375	375
Defective				386
Crack	14	-	4	18
Microcrack	192	-	48	240
Finger inter.	93	-	24	117
Black spots	-	4	3	7
Bad Soldering	-	2	2	4

Table 6.3: Sample distribution in the dataset

In order to check the impact of every imprinting, after every imprinting operation, the network was executed on the entire test set. Some samples from these executions are shown in Figures 6.10 and 6.11. The samples in Figure 6.10 illustrate samples with the base classes (cracks, microcracks, and finger interruptions), while the samples in Figure 6.11 show some instances of the new classes (i.e. bad soldering on the left cell and black spots on the right cell).

Figure 6.10 shows that the base network (i.e. U-net trained on base classes) segmented the base classes with high precision. As expected, the segmentation was more accurate when the defects were bigger and present higher contrast with respect to the cell background. This is especially visible in the finger interruptions in the third sample, where the darker instances are more thoroughly segmented than the lighter ones. With respect to the samples with the new classes in Figure 6.11, the black spots instances were not segmented in any of the samples. In the case of the bad soldering class instead, the network very vaguely detected some defective features in the top-right corner of the cell that belongs to a bad soldering defect.

After the first imprinting (i.e. black spots class addition), the instances from black spots were segmented as can be appreciated in the first and second samples in Figure 6.11. In this case also, the darker the defect, the better the segmentation was. However, the imprinting affected the segmentation of the base classes as can be seen in samples from the second row in Figure 6.10. The network segmented some pixels at the borders of the "older" defects as they belonged to pixels from a black spot class, for example, the microcrack in the second sample.

After the second imprinting with the bad soldering class, the same effect as the first imprinting was appreciated. The network was able to segment the bad soldering instances

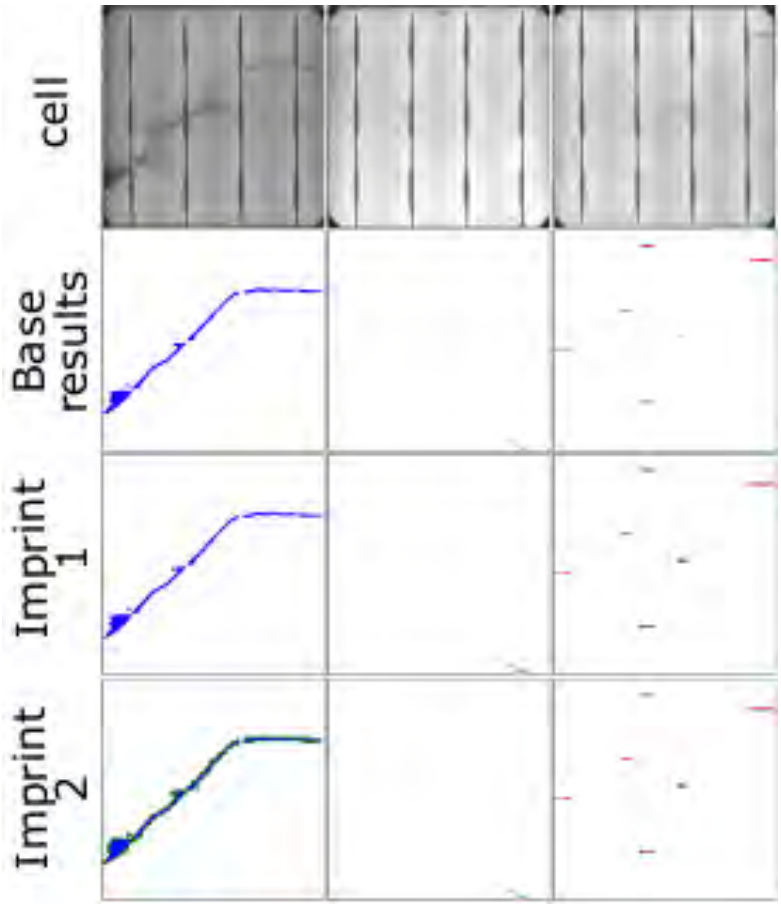


Figure 6.10: Segmentation results on samples with base defect classes before imprinting and after each imprinting. The colors of the classes are: blue-cracks, light green-microcracks, red-finger interruptions

as can be observed in the last two samples in Figure 6.11, but at the same time, it implied the appearance of some false positives around the previous segmentation results. Nonetheless, as it is clearly visible, the segmentation of the bad soldering was poorly accomplished.

Overall, the weight imprinting allowed the network to segment new classes using just a few defective samples from each new classes (2 for bad soldering and 4 for black spots). Nevertheless, after the imprinting, the network started mistakenly classifying certain areas around the "older" defects as they were new defect class instances.

In addition to the qualitative results, quantitative metrics were also computed to also illustrate the effect of the imprinting on the model performance. The results for these metrics are in Tables 6.4 and 6.5. As with the segmentation results, the results in

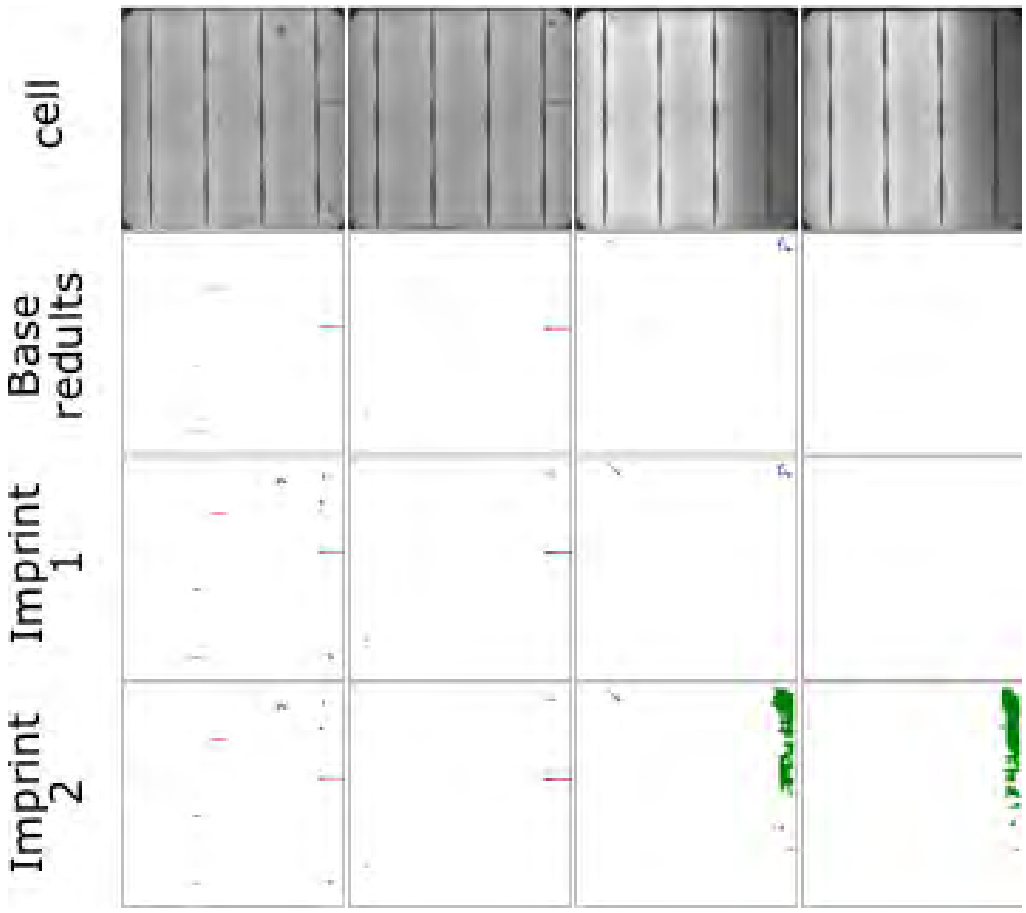


Figure 6.11: Segmentation results on samples with new defect classes before imprinting and after each imprinting. The colors of the classes are: brown-black spots, and dark green-bad soldering.

the tables represent the performance with the base network, and after each imprinting operation.

	Recall	Precision	Specificity
Base network	88	86	99
1st Imprint.	96	81	96
2nd Imprint.	97	77	94

Table 6.4: Results at image level before and after each imprinting.

As can be observed, in general terms the imprinting allowed the network to detect more defects within the cells, and thus, detect more defective cells making the Recall increase after each imprinting operation. Also, the imprinting made the network segment some instances of the base defects that were initially very vaguely segmented, turning some

	Orig. model (FCN8)	Base model (Unet)	Imprint. 1	Imprint. 2
Crack	100%	100%	100%	100%
Microcrack	64%	71%	90%	90%
Finger inter.	65%	88%	90%	90%
Black spots	0%	0%	77%	77%
Bad Soldering	0%	0%	0%	100%

Table 6.5: Percentages of detection of each model per defect class.

False Negative into True Positive cases. If the results are broken down by defect classes as shown in Table 6.5, the effect of the imprinting is easily appreciable in the case of the new classes, but also in the case of samples with microcracks. In the case of the latter, the models went from detecting about 71% of the microcrack instances to around 90% of the samples.

The improvement in microcracks is given by those instances that are very small and in some cases resemble black spots. In these cases, previous to the imprinting, the network classified the pixels corresponding to the defect as background, thus they were considered as undetected. But after the imprinting, these pixels were classified as black spots and microcracks. From the defect detection point of view, as the main objective is to highlight the presence of defects in the cells, the defect instances were considered detected as reflected in the results. However, from the pixel segmentation point of view, as mentioned when describing Figure 6.11 the pixel classification got a bit noisy after the imprinting operations.

In addition, the network started to segment certain areas in defect-free samples as they were defective (mainly dark areas that resemble black spots). After a manual analysis of these cases, it was found that some of them could be classified as defective as they really contained these black spots, in others instead, they were just False Positive cases. In total from the 375 defect-free samples, there were 15 samples that could be considered as defective based on the presence of black spots. Taking this into account, these 15 cases were put aside and the remaining 360 defect-free samples were employed for the metrics computation. As mentioned, the imprinting operation makes the model start to segment certain areas within the defect-free samples as they were defective areas. This translates to more False Positive cases making the Specificity as well as Precision decrease as reflected in Table 6.4 and Table 6.5.

6.3 Concluding remarks

In this chapter, two alternatives with which the models can be adapted to new scenarios in the production line using few defective samples have been described. These techniques can be included in what could be denominated as the adaptation stage, which would be the third and last phase of the proposed methodology. In this way, the methodology would consist of three phases in which inspection models would be obtained and improved to adapt them to the production line.

The techniques that were explored were two: Transfer learning, where the models can be adapted to different data domains using the weights from trained models as a kind of specialized initialization. And Few-shot incremental learning, where new defect classes can be sequentially incorporated into a model using just a few representative samples from those new classes.

In order to check the applicability of the techniques as model adaptation techniques in the methodology, several experiments have been carried out in the context of quality inspection of solar cells.

In the case of transfer learning, three experiments have been conducted where models trained on one type of solar cells have been adapted such that they could work on other types of cells. The first experiment focused on adapting models from working on monocrystalline cells to working on polycrystalline cells. In the second experiment instead, the objective was the opposite adaptation, i.e. from polycrystalline to monocrystalline. And in the third experiment, the adaptation consisted in adjusting models trained on polycrystalline cells with 3 and 4 buses to be applicable on cells with 5 buses. The results from the experiments have shown that Transfer Learning can be a tool to take advantage of already trained models to quickly obtain models for different production lines. With few defective samples, it is straightforward to adapt inspection models from other production lines to new lines, and thus, speed up the commissioning process.

In the case of Few-shot incremental learning, a base network was trained to detect three types of defects, then two additional types of defects have been incorporated achieving a model capable of detecting 5 classes of defects. The results from the experiments have shown that the few-shot incremental learning technique has the potential as a model adaptation tool. It has been shown that it can be employed to incorporate new defects into the models and keep them updated across the life cycle of the production line. However, it has also been seen that each time a new defect has been incorporated,

the results at the pixel level have turned noisier, which should be improved in future iterations of the work with this technique.

Chapter 7

Methodology deployment

7.1 Introduction

The previous three chapters have introduced several DL based techniques to build an industrial inspection system. Each technique has focused on a different scenario that a production line will commonly face during its life cycle. The anomaly detection based approach, described in Chapter 5 focuses on the commissioning of a new production line where few defective samples are usually available for a proper supervised training. Because of that, the use of defect-free samples is proposed in order to obtain an initial inspection model. Then, in Chapter 4, supervised learning techniques have been described which are thought to be used in a more advanced stage of the production line where defective samples will be more abundant. And lastly, model adaptation techniques are described where the objective is to try to adjust already trained models to changes that may occur in the production line, like the need for detection of new defects or the need to handle slightly different cells.

Each technique is defined to solve different problems in production, however, the approaches can be understood as different building blocks that together can define a methodology to build an inspection system as illustrated in Figure 7.1. Each block tackles specific circumstances and can be applied independently from the others, nonetheless, the output from one block can be considered as the input for the next block. In this way, a methodology can be outlined where the techniques are applied in a sequential

manner from the beginning of a production line to then smoothly, adapt the models as the production line evolves.

- *Do we have defective samples?*
  - *No →* ***STAGE 1 – Unsupervised + Anomaly detection***
  - *Yes*
    - *Do we have labels?*
      - *No →* ***STAGE 1 – Unsupervised + Anomaly detection***
      - *Yes*
        - ***STAGE 2 – Supervised training*** *(*to get a model)*
        - ***STAGE 3 – Model adaptation*** *(*if we want to update the model)*

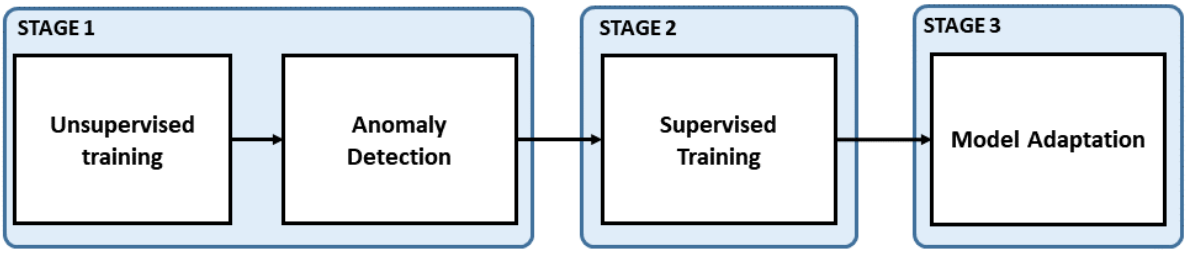


Figure 7.1: Diagram on how could be applied the proposed methodology. Note that this schema is a simplification of the schema in Figure 1.3.

This chapter is going to focus on describing how the different building blocks can be used within a methodology applicable to many quality inspection industrial cases. Also, an experiment will be described with which the feasibility of its application in the context of industrial solar panel production will be proved.

7.2 Methodology outline

Before starting with the description, it should be remarked that even though the steps in the methodology are thought to be sequentially applied, is by no means the only way to do it. The premise from which the methodology was designed consists in a scenario where there is neither an already trained model nor defective samples available to start from. However, this might not always be the case, sometimes defective samples or trained models might be already available from the very beginning of the process. Apart from that, the methods that were chosen are not the only options available in the literature. The practitioners have access to a vast amount of network architectures, modules, approaches... in the literature to choose from to reach the same point described

in the preceding sections. In this sense, the methodology described just outlines the steps that were found interesting to follow to build an inspection system. Therefore, practitioners can safely skip stages and employ different techniques if they find them more convenient or match better their use of case.

Nonetheless, the description below consists in a summary of how the stages in the methodology were thought to be applied in order to build a solar panel inspection system from scratch.

Initial stage - anomaly detection: As stated above, at the beginning of the deployment of a new production line there is commonly a scarcity of pre-trained models or defective data samples that can be leveraged to build an inspection system. However, in these cases, there is usually plenty of defect-free data samples to access. Taking into account such scenario, it would be interesting to take advantage of these defect-free samples and try to build an initial inspection model using the anomaly detection approach.

In Chapter 5, the architecture called f-AnoGAN was described as one of the possible network architectures that can be used to build this initial anomaly detection model. The training of this network aims to make the network learn the probability distribution of defect-free samples, such that, the network will only be able to encode and reconstruct those samples that closely follow the defect-free distribution. This reconstruction based anomaly model can then be used to process new upcoming data and detect anomalous samples as well as locate anomalous areas within the samples. Moreover, the anomaly location feature can be exploited and turn the anomaly model into an automatic annotator as can be seen in Figure 7.2, and thus, used it to build an annotated defective dataset with the upcoming defective samples without any major human intervention. In this way, the first stage of the methodology can provide the practitioners with an inspection model to start inspecting cells in production from the very beginning phase of the production line life cycle and also automate the data labeling process that will come in handy in the subsequent methodology stages.

Second stage - supervised learning: Once the anomaly detection model is deployed, it will start separating samples into anomalous and no anomalous classes, as well as identifying anomalous areas within the samples. After a while, the production line will have produced enough defective cells making it possible to build a dataset containing enough representative defective samples for an adequate supervised training. As shown

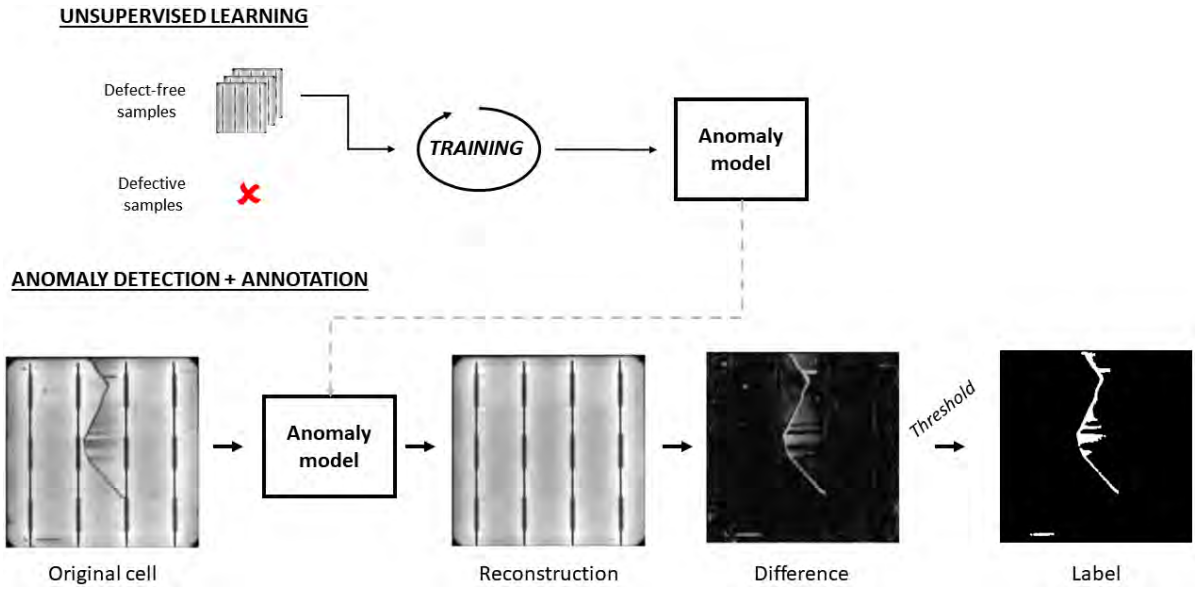


Figure 7.2: Basic diagram on how the automatic labels are obtained.

in Chapter 5, this procedure not only outputs a model that improves the results from the anomaly model, but also can reach a similar outcome as if the model was trained on laborious manual labels. However, the training of the model using this procedure can be greatly automated and can be used to avoid the time-consuming data annotation task.

The idea at this stage is to take advantage of the annotation capabilities of the anomaly model and employ it to automatically generated annotations to train a supervised model to reach higher rates of accuracy. In this way, the initial two stages of the methodology can be connected and a smooth transition from the first stage to the second can be defined. This procedure will not require major human intervention. The only effort that the operator will need to do is to review the automatically generated labels and refine them if necessary to reach the most accurate supervised model possible.

In Chapter 4, the reader was provided with two possible alternatives for conducting the supervised training stage. One technique followed the sliding window processing schema where the images were processed by patches using a very simple CNN architecture. Instead, the second technique focused on directly outputting a dense prediction after one forward pass of the network using a Fully Convolutional Network architecture. As mentioned previously, these options are not the only ones available in the literature that can fulfill the purpose of the second stage in the methodology. However, in order to take advantage of the pixel level labels coming from the anomaly model, it will be appropriate to limit to those approaches that somehow work in the same way as the

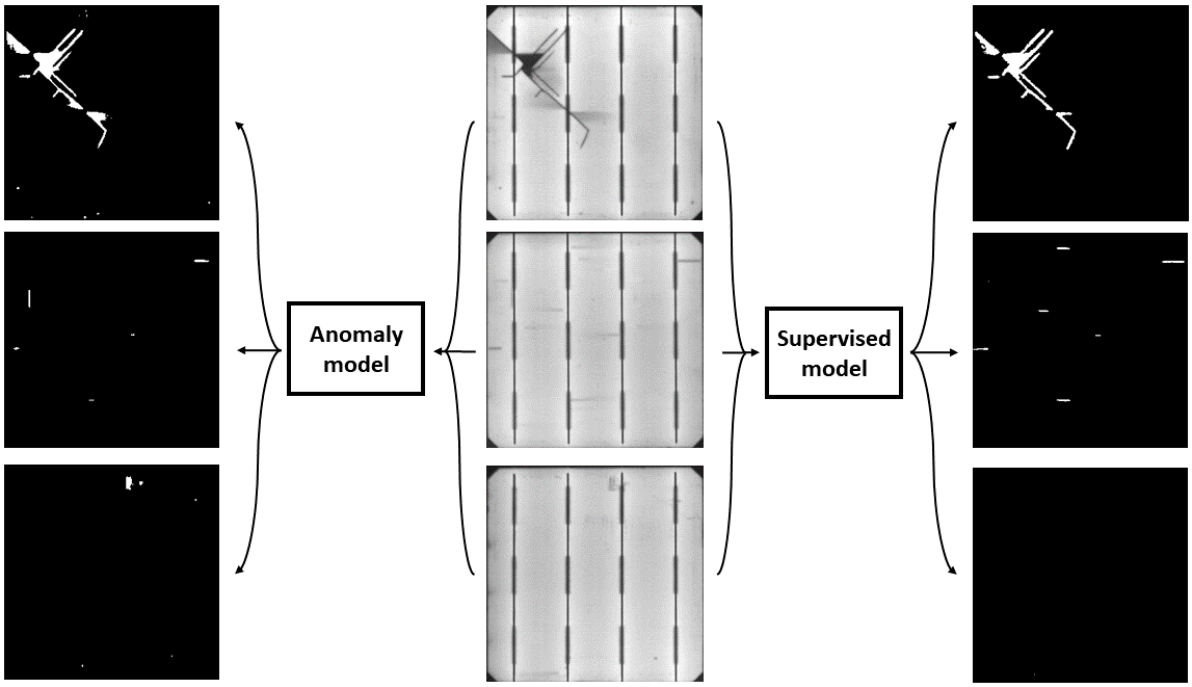


Figure 7.3: Results example from supervised and anomaly detection models. The supervised model is able to keep the segmentation results obtained from the anomaly model regarding the defects, but at the same time reduce the number of false positives related to noisy or dirty areas in the cells.

explored architectures as pixel level predictions can improve the interpretability of the results.

Third stage - model adaptation: Finally, after the anomaly model and supervised model are trained and deployed, they will be employed to both accurately inspect the production line and detect new anomalies that will appear from time to time. As the production continues, it could happen that the operator starts seeing that the anomaly model is pointing out new types of defects in the cells, maybe because of a new material that has been lately incorporated into the product, or maybe because of some parameters or configuration in the production has been changed. It might also happen that the manufacturer realizes that detecting the new feature in the production could help to improve the overall product quality.

In this case, the adaptation techniques explored in Chapter 6 (i.e., Transfer Learning or Few-shot learning) focus on adapting the supervised model to these new features. Note that in the case of selecting the few-shot learning, the network architecture should follow the architecture specified in the model adaptation Chapter 6. Otherwise, the original

U-net architecture will lack a mechanism to perform the weight imprinting as specified in the model adaptation Chapter 6. Also, in the experiment carried out, the defects were required to be grouped in classes in order to perform a multiclass segmentation to then add new classes. The anomaly detection does not provide an anomaly classification output but a binary classification (defect-free sample vs anomalous sample). In order to perform that multiclass segmentation, this gap should be filled by manually clustering the anomalies, for example, by size, location, or other kinds of features.

7.3 Experiments and Results

In order to show how the proposed methodology would work, an experiment that simulates the first two stages in the methodology (i.e., anomaly detection and supervised learning) was performed. The simulation consists in a scenario where there is neither a trained model nor defective samples. Therefore, the anomaly detection approach will be the initial approach to be used to then use these anomalies as automatic labels to train a supervised model and reach an improved inspection model. This experiment was thought of as a way to show the effectiveness of the proposed methodology by identifying how many defective samples could be detected compared to a supervised model trained on manually annotated data samples. For such a purpose, the unlabeled dataset composed of 700 monocrystalline solar cell panels described in Section 2.4.4 was employed.

First, following the methodology, the just defect-free cells were selected from the first panels. For this step, the f-AnoGAN with the included modifications was employed as described in Chapter 5. Regarding the training samples, 750 defect-free cells were employed as was done during the experiments in Chapter 5, which corresponds to about 12 panels. After that, the anomaly detection model was executed on all the cells in the dataset to automatically detect and annotate the anomalous samples among all the cells.

Once the cells were separated as defective and non-defective and annotated at the pixel level, four different supervised models using different amounts of defective cells were trained. In this case, the same configuration of the U-net network described in Chapter 4 was employed as the network architecture for all the experiments. The only variation between the models consisted in the number of samples in the training dataset.

The first model was trained with the samples from the first 20 panels with defective cells (the first 20 panels with defective samples are within the first 35 produced panels). From the first 1200 cells (20 defective panels $\times$ 60 cells) the defective samples that

were detected were 69, from where 6 contained a crack, 12 microcracks, and 67 finger interruptions. Note that the numbers do not sum up to the total. This is because almost all the defective samples, the ones with cracks and microcracks also included finger interruptions, but the inverse scenario does not hold. This must be clarified as it will be repeated in the following experiments as well. Then, the second model was trained with the same 69 defective samples used in the first experiment plus the defective samples from the following 20 panels with defective samples in the panel sequence (the total sequence was 78 panels). In the first 40 panels, 157 defective cells were detected, from which 8 samples contained cracks, 28 contained microcracks, and 141 contained finger interruptions. Then the following two experiments were done with the first 80 panels and first 100 panels with defective samples which were extracted from the first 148 panels and first 190 panels. The defective samples for the latter experiments were 298 (17 cracks, 75 microcracks, and 256 finger interruptions) and 392 (21 cracks, 102 microcracks, and 339 finger interruptions), respectively.

Once the models were trained, they were executed over the remaining sequence of panels. The chart in Figure 7.4 illustrates the accumulated defective cells detected by each model across the sequence of the panels. To count a sample as defective, the prediction by the models must have over 40 pixels. This threshold was set taking into account that the smallest predictions in the results were regarding small finger interruptions defects that had about this size and from our perspective were good enough to be considered as good detection. From now on, for the sake of making the reading easier, the models obtained from these experiments will be referred to as panel20, panel40, panel80, and panel100.

Note that when plotting the results, the initial values in the experiments with automatic labels were set the same as the ones from the model trained on manual labels. In other words, in the case of panel20, the first 35 results (35 panels) were set equal to the first 35 results from the model trained on the manual labels. This was repeated with the other models too. This was done because in each experiment the initial panels were used as training samples. Any model had either too many false positive cases or too many false negative cases. The only model that showed a slightly different amount of detected defective samples was the model panel100, which showed lower detection rate capabilities.

As can be seen, this latter model seems to detect many more defective samples than the other models, however, within these samples a great chunk could be considered as false positive cases.

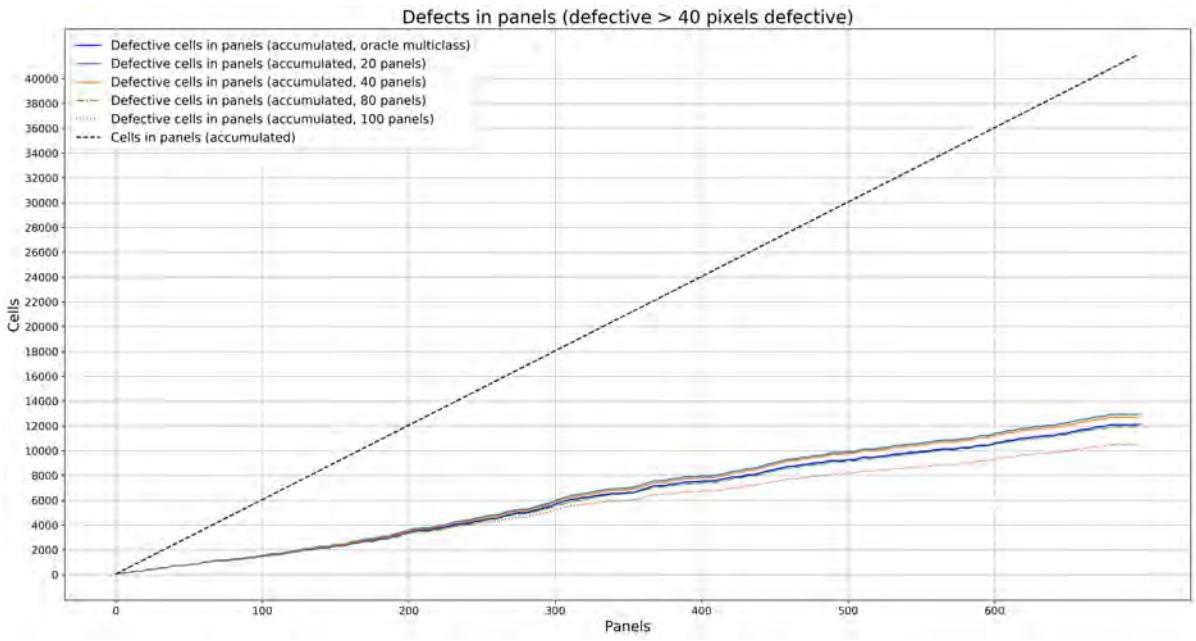


Figure 7.4: Results from the model trained with manual labels and models trained with automatically generated labels.

In total, the supervised model trained on manual labels was able to detect 12,234 defective cells out of 42,000 processed samples. In contrast, the experiment with automatic labeling detected 13025 in the case of model panel20 (6% more), 12,814 in the case of the model panel40 (4% more), 12,049 in the case of the model panel80 (2% less), and 10,630 in the case of the model panel100 (12% less). Despite the differences, the detection of severe defects like cracks and major microcracks was not much different among models. All of them (including the model panel10) were able to detect all the samples with cracks (77 samples), and the majority of the samples with medium-big size microcracks (405 samples) like the ones illustrated in Figure 7.5. In the latter defect class case, the detection rates were 96.3% in the case of model panel20, 97% for model panel40, 96.8% for model panel80, and 90% for model panel100. The differences in the results that can be appreciated in the chart come from the detection of finger interruptions that consisted in the defect class with a greater amount of instances (almost 90% of the defective cells had some sort of finger interruptions), and from some samples with peculiarities that were considered as defective by the models. In this sense, one of the conclusions that can be drawn from the results is that does not matter the number of defective samples used for training. The models with few defective samples can detect the most severe and crucial defective samples as well as the models trained with more samples.

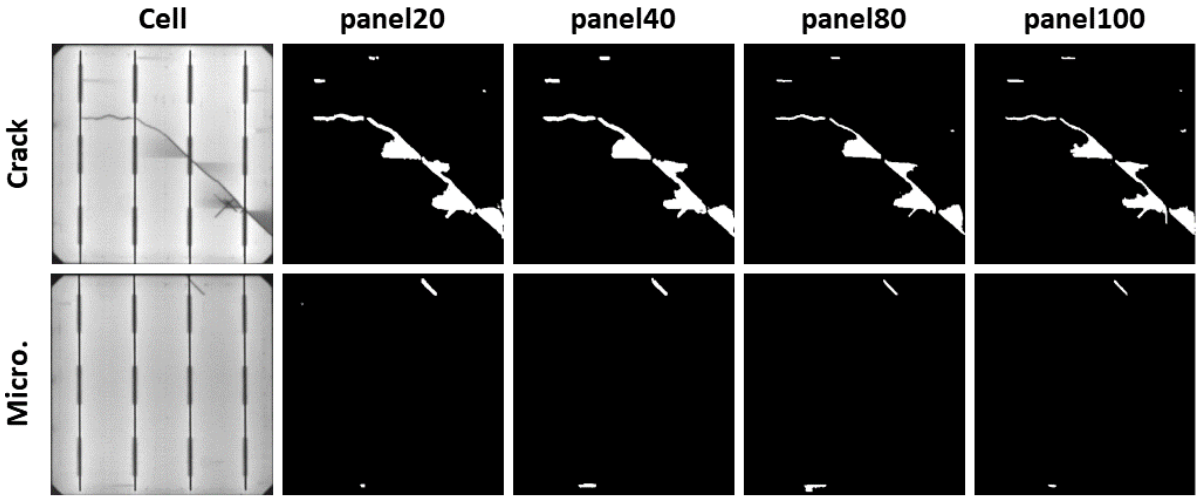


Figure 7.5: Examples of the results of the different models on a crack and a micro defect classes.

Regarding less severe defective cells, almost every defective sample in the dataset that was used for this section contains some sort of finger interruption defect class instance. Some of these samples contained severe types of finger interruptions like the one exhibited in the first row in Figure 7.6. In this case, none of the models had any problem detecting them as they presented a big gradient with respect to the background. Nonetheless, some samples contained a very light type of finger interruption cases like the one illustrated in the second row in Figure 7.6. In this latter case, the behavior of the models was not that homogeneous. This is one of the primary sources of the differences shown in the chart.

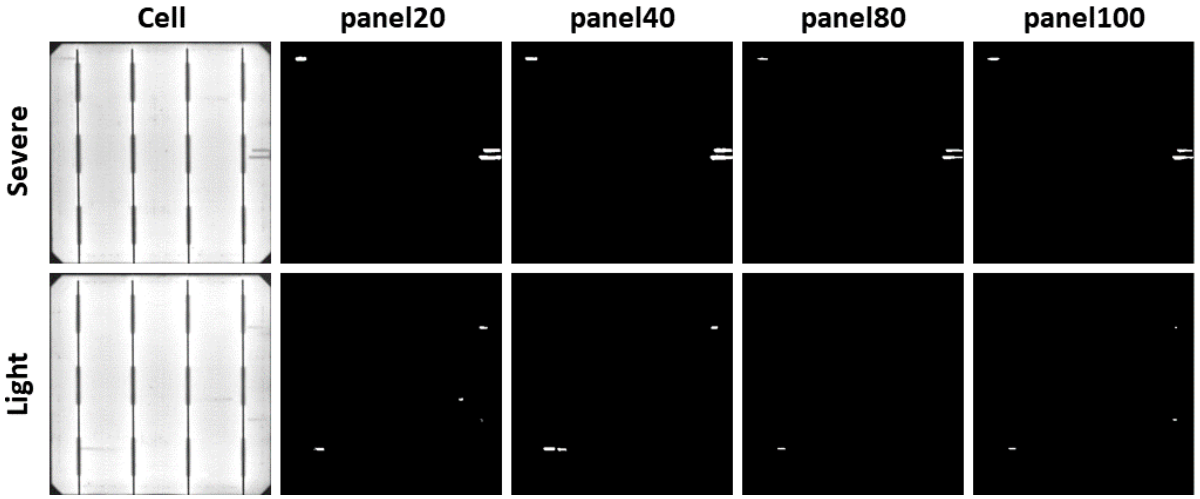


Figure 7.6: Examples of the results of the different models on a severe and light finger interruption defect class.

The other reason for the differences in the results are the samples like the one exhibited in Figure 7.7 with kind of dirty looking areas in the cells. Some cells present dark areas around the edges may be because some dirt interfered during the EL image capturing or because there was a problem when capturing the images. These areas have been sometimes detected as defective, and thus, have increased the overall amount of defective samples detected.

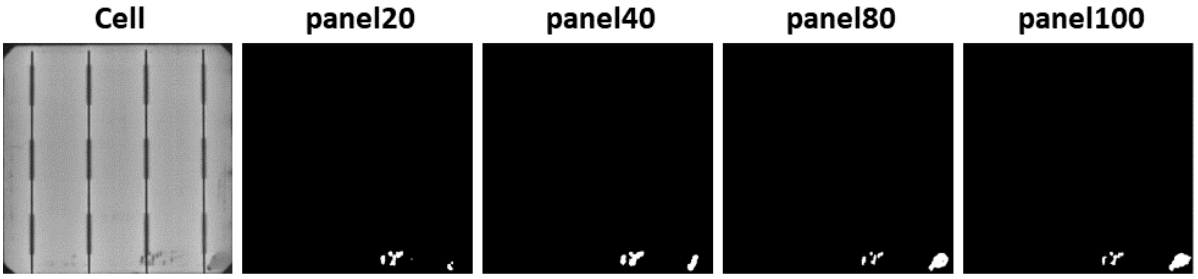


Figure 7.7: Example of the results of the different models on a sample with noisy area.

It is believed that one factor that might have affected the results could be the balance of defect classes in the training set, and correlated to that, the number of defective pixels over the background pixels. In the training of model panel20, 6 samples with cracks out of a total of 69 samples were employed. Although the cracks can be considered relatively big compared with the size of the cell, the number of defective pixels compared to the overall number of pixels in the cell is not that much. This can challenge the model during its training. This is more accentuated when contemplating the microcrack and finger interruption samples where the difference is even greater. As new panels have been considered as part of the training process, more microcracks and finger interruptions have been included and not so many cracks. I believe that increasing the unbalance between defective and background pixels has had an impact on the results.

7.4 Concluding remarks

This chapter has outlined how the techniques described in the previous chapters can be understood as independent building blocks that when put together, they define a methodology that can serve to build an inspection system. Each block tackles a different scenario that production lines face during each life cycle making the most out of the available resources at each time. Nonetheless, as illustrated in Figure 1.3, there are links between the blocks that show how the output from one block can constitute the input to another block. In this chapter how these links could be followed and sequentially

employ the techniques in each block to obtain inspection models and refine them has been explained. In addition to describing the methodology, an experiment has also been carried out to show its applicability in an industrial case of quality inspection and to show what has been obtained in the process.

Chapter 8

Conclusion and future works

This chapter contains the final remarks about the thesis and also outlines some of the possible future work lines that could be followed based on the limitations that were during the thesis.

8.1 Conclusions

This document can be concluded by underlining that this thesis has described a three stage based methodology that provides industrial practitioners with several techniques that can serve as guidance to develop a DL based inspection system. The different techniques focus on different stages that a production line can go through during its life cycle so the practitioner can choose which technique can use in each case. All the works that have been reported here have been focused on the solar panel production scenario, but it is believed that should be extendable to other domains.

Before starting with the experiments, a literature review was carried out to identify the proposals made in the field of quality inspection of solar cells. During the review, the approaches were grouped into three categories based on the amount of human intervention the techniques required for their application. From the three categories, the thesis focused

on DL based methods as they present higher levels of flexibility than more traditional approaches as stated in the state of the art chapter.

In Chapter 4, two supervised learning based techniques that worked fine with few annotated data samples and also provide segmentation results for pointing out defective regions are described. The first technique consisted in a Convolutional neural network designed for classification which is mixed with the well known sliding window processing scheme. By mixing these two concepts, the network can yield defect heatmap like results that serve as defect location proposals for inspection. This approach was able to detect most of the defective samples in the cells, but being subject to a somewhat lengthy processing may face difficulties from the perspective of its actual application. On the other hand, a Fully convolutional neural network approach is explored which allowed us to process the entire image at once. This procedure notoriously decreased the processing time and still provided us with a segmentation of the results.

The supervised learning approaches are accurate techniques as long as defective labeled data samples are available for training. This requirement can not always be ensured, and even less in a scenario like the industrial quality inspection. In order to overcome such limitations, in the following chapter 5 the anomaly detection approach is explored. In this approach, the training set is not composed of defective data samples but of defect-free samples which are usually more accessible in industrial setups. In this way, practitioners can avoid waiting to have defective data samples for training as they would have to do with the supervised approach. For this second approach, a Generative Adversarial Network was employed, more specifically f-AnoGAN network. In addition, some modifications were included in the network which resulted in defect detection rate improvement compared with the original network proposal. The motivation behind this network consists in training it to learn the probabilistic distribution of the normal data (i.e. defect-free samples) through the data encoding and reconstruction training scheme. This scheme will force the network to only be able to reconstruct defect-free data no matter the input data, which will allow us to detect defective cases as the difference between the input and reconstruction will be bigger than in the case of defect-free cases.

Apart from its application as a defect detection and location model, its application as an automatic annotation generator for supervised learning is also explored. The feasibility of such application was tested by comparing the results between a model trained on a manually labeled dataset and the same model but trained on labels automatically generated by the anomaly model. The experiment showed us that the results are similar to each other, and in both cases, more precise than the anomaly model. Thus, it could be

considered as a feasible approach to improve the anomaly model accuracy while avoiding the usual manual labeling task.

Then, in chapter 6, different alternatives for model adaptation are explored. Once the inspection models are deployed in production, they may encounter cases for which they have not been trained and should be adapted to cope with. In order to adjust the models to these kinds of situations two techniques are described. On the one hand, the Transfer learning technique was tested. This technique is a well known technique that has already become the way to go when trying to train neural networks and there are not many data samples for it. The testing of this Transfer learning consisted in employing the technique in three different scenarios where models trained on one type of cell were adapted such they could work on another type of cell. On the other hand, a more recent technique that lately has been gaining track called Few-shot learning was explored. This technique focuses on trying to increment the number of classes that a trained model can work with once it has been trained. To test this technique, an experiment was conducted where an initial base model was first obtained, and then the base model was adjusted such that it could work with additional classes. The initial base model was trained to work on three different defect classes, and then, the sequentially two additional defect classes were incorporated such that a final model that could work with five defect classes was obtained. As for the model, the backbone structure of the network from the original work was adjusted resulting in an improvement in defect detection with respect to certain small defect classes.

Lastly, a final experiment was carried out where the scenario of a company that wants to start with a new production line and does not have either a previous model or defective samples for a supervised training was emulated. Thus, how the proposed methodology could be applied and what kind of results could be obtained on a dataset composed of 700 monocrystalline cell panels is described. The emulation started with the anomaly detection approach and then turned to the supervised approach taking advantage of the results from the anomaly model. In the supervised training, a comparison between the results using a model trained on manual labels and the results from several models using different amounts of defective cells extracted from different amounts of sequential panels is performed.

This thesis primarily focused on DL methods to define methodology mainly motivated by the flexibility these methods provide in dynamic industrial scenarios compared to the more traditional methods. Nonetheless, traditional methods are still very valuable when designing inspection systems. It is thought that both DL and traditional techniques

should be part of the brainstorming process when designing a robust inspection system. This should be emphasized to avoid conveying the idea that DL techniques are always the way to go and that more traditional techniques should always be discarded.

In addition, among the different neural network architectures available in the literature, just the ones that were found suitable for interpretable defect detection under a data scarce industrial scenario were selected. Nonetheless, by digging a little into the field of DL, one can easily realize how many variants of neural networks exist and can be used for the same purpose. The objective of this thesis was to outline a methodology that practitioners could use when designing the inspection system. In this sense, practitioners could, and should, also explore other architectures in order to pick the optimal for each use case.

Finally, it should be also remarked that even though not mentioned in its corresponding section across the document, some of the techniques described have been already incorporated or are in an advanced phase of being incorporated into real industrial quality inspection solutions. This might not be a great contribution, but the point of seeing that the techniques explored in the thesis have real industrial applicability is another point that is believed that should be underlined.

8.2 Suggestions for further research

During the development and evaluation of the different methods, several limitations and improvement areas were identified. In the following, some of the ideas that are believed that should be addressed as the following steps of the works in this thesis are going to be enumerated:

Evaluation metric: the main objective of the quality inspection is to detect defective samples among all samples that are being produced. If this is done through images, it can be translated as image classification. In this thesis, apart from focusing on the classification task, the methods employed have also been focused on image segmentation to improve the interpretability of the results. In the case of evaluating classification, there has not been any problem. In our case, an arbitrary threshold was taken with respect to the number of defective pixels detected in the samples to decide whether the samples should be considered defective or non-defective. Once this is done, the objective evaluation of the models is pretty straightforward to perform using the already available

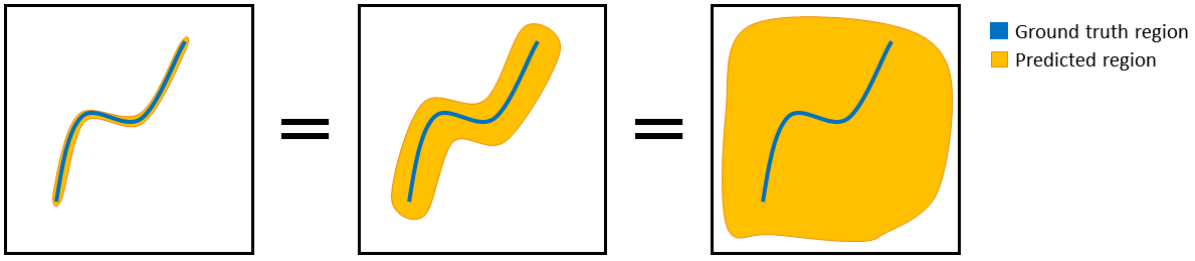


Figure 8.1: Three different prediction cases where the PRO metric will output a perfect score.

metrics. The criteria to decide if a cell is defective or not can be done in this arbitrary way, or also by training the network to output this assessment directly.

In the case of segmentation, as perfect segmentation results were not sought but just reasonable defect segmentation, a qualitative assessment was thought to be sufficient to describe results. However, a qualitative evaluation is not always enough and a quantitative assessment should be pursued. In image segmentation related problems, there are different available metrics to numerically analyze the results. For example, in the literature, the Intersection Over Union (IoU) is quite an extended metric to provide an evaluation of the results. There are also other region-based and contour-bases metrics like Trimap[58], Hausdorff distance [30], per-region overlap (PRO) [13], Saturated Per-Region Overlap (sPRO) [14], or the ones in [31, 39] that go a step further and consider the predictions as blobs (i.e., same class groups of pixels) to try to assess whether the object in the ground truth can be considered detected or not. These latter, [31, 39], are for overall image score using boundaries.

In the case of IoU, it was employed as an evaluation metric, but in our opinion is quite severe at penalizing slightly over-segmented areas. In our case, the defects in the cells are very narrow structures, thus a slightly over-segmented prediction, even following the structure of the defect and considered good quality by human operators, can yield very low evaluation results using these metrics. In the case of the other metrics, they mainly contemplate 1-vs-1, prediction vs ground truth blob scenario. In some of the predictions obtained in this thesis, the scenarios that were found were more like N-1 prediction vs ground truth cases. Instead of a unique prediction, this can be composed of multiple blobs that resembled the real defect but that are not connected to each other. In this case, it was not found how to proceed specifically with the analysis. It is not so straightforward to attribute a predicted region to a True-positive or False-positive classification. From a human operator's point of view, the analysis would be easier to do like the overall shape of these groups of blobs would follow the real defect shape.

In [8], a custom metric to evaluate individual ground truth regions that was very similar to the PRO [13] and sPRO [14] metric was tested. [13] evaluates the prediction by computing the average of the overlap between the regions in the ground truth and the prediction. In other words, they evaluate whether the ground truth regions are overlapped with the prediction and compute the average of all the overlapping values. However, do not consider the over-segmentation cases, i.e., the case where the prediction does not only overlap the area of the defect but exceeds it considerably as represented in the third image in Figure 8.1. If a prediction consists of the entire area in the evaluated samples as it is defective, the PRO metric will still output a perfect score since the ground truth area is completely covered by the prediction. It is believed that the region coverage is a good way of evaluating the detection, but it has to also take into account that the prediction does not take too much of such area. If not, very bad predictions like the excessive overlapping case can go unnoticed. In our proposal, these excessive overlapping cases were taken into account but the results were not satisfactory enough. It was hard to equally or similarly evaluate those predictions that resemble almost perfectly the defect shape and those where the overlap rate was a bit higher even though the prediction shape was fine.

Apart from that, in our case and in the case PRO metric is still difficult to attribute the predictions to a True-positive or False-positive classification. For example, if the coverage is not good enough, the regions that partially overlap the ground truth region should be evaluated as one FP or each prediction region should be considered as independent FP cases?

Taking this into account, one of the future works identified in the thesis consists in designing an analytical and systematic way of evaluating the segmentation results that somehow will resemble how humans will assess if the segmentation blobs should be considered as good detection or not. A more objective way of evaluating the objective, but at the same encode human perspective, will allow us to numerically compare models' performance.

In the case of not being able to find such a metric, based on the experience during this thesis, it would be more interesting to stick with the bounding box based object detection concept as the way to go for defect detection, and add a segmentation layer over to improve the results interpretability. The bounding box offers an easier way of objectively analyzing the results using the IoU metric against the ground truth bounding box as well as the Non-maxima suppression technique to stick with one prediction instead of a bunch of blobs.

Search for trends in production: The results from the inspection systems can reveal more information about the production than just the status of the produced parts. By storing and analyzing the results from the models, it could be possible to detect and forecast trends in production, or also get more information about the status of the steps prior to the inspection.

For example, in the context of solar panel inspection, the inspection system may start to detect defects repeatedly in one of the cells on the edge of the panels. This can be indicative of something that is not working well in production. by evaluating the results holistically (at the panel level), it might be possible to go back in the process and search for the source (or sources) of failure that has caused the defects to appear. Also, by storing historical information about the quality control results, it might be possible to identify the moment or circumstances that led the production line to produce the defective cells. In this way, the occurrence of defective cells could forecast and avoid them.

More configurations: As stated in the previous section, a selection of DL methods that complied with the purposes of defect segmentation as well as worked with a few defective data samples was performed. This selection is by no means the only possible choice of methods that could be made. The main idea of the thesis was to outline a methodology that industrial practitioners could follow to design an inspection system based on DL methods. Regarding this aspect, other kinds of network architectures (other base architectures, additional modules..), different training strategies, a mixture between DL methods and more traditional approaches, different losses, and an endless list of possibilities could be explored to find the best solution for every specific case. Some of the possible ideas to implement over the architectures in this thesis could consist in:

- Adding attention modules to the network which seem to improve the original baseline network results. Specially, if they can help with small defects like microcracks or finger interruptions.
- Adding a classification branch to the networks (mainly U-net) such that, in addition to the segmentation output a classification output will also come from the network. In this way, it will avoid the need of defining an arbitrary rule to decide whether the processed piece is defective or not.

Bibliography

[1] Akram MW, Li G, Jin Y, Chen X, Zhu C, Zhao X, Khaliq A, Faheem M, Ahmad A (2019) Cnn based automatic detection of photovoltaic cell defects in electroluminescence images. *Energy* p 116319

[2] Akram MW, Li G, Jin Y, Chen X, Zhu C, Ahmad A (2020) Automatic detection of photovoltaic module defects in infrared images with isolated and develop-model transfer deep learning. *Solar Energy* 198:175 – 186, DOI https://doi.org/10.1016/j.solener.2020.01.055, URL http://www.sciencedirect.com/science/article/pii/S0038092X20300621

[3] Anwar SA, Abdullah MZ (2014) Micro-crack detection of multicrystalline solar cells featuring an improved anisotropic diffusion filter and image segmentation technique. *EURASIP Journal on Image and Video Processing* 2014(1):15

[4] Arjovsky M, Bottou L (2017) Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:170104862*

[5] Arjovsky M, Chintala S, Bottou L (2017) Wasserstein gan. *arXiv preprint arXiv:170107875*

[6] Baier S (2012) Lumenistics - what is full spectrum lighting? http://lumenistics.com/what-is-full-spectrum-lighting/

[7] Balzategui J, Eciolaza L (2022) Few-shot incremental learning in the context of solar cell quality inspection. *arXiv preprint arXiv:220700693*

[8] Balzategui J, Eciolaza L, Arana-Arexolaleiba N, Altube J, Aguerre J, Legarda-Ereño I, Apraiz A (2019) Semi-automatic quality inspection of solar cell based on convolutional neural networks. In: 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Zaragoza, Spain, pp 529–535, DOI 10.1109/ETFA.2019.8869359

[9] Balzategui J, Eciolaza L, Arana-Arexolaleiba N (2020) Defect detection on polycrystalline solar cells using electroluminescence and fully convolutional neural networks. In: 2020 IEEE/SICE International Symposium on System Integration (SII), IEEE, Honolulu, HI, USA, pp 949–953, DOI 10.1109/SII46433.2020.9026211

[10] Balzategui J, Eciolaza L, Maestro-Watson D (2021) Anomaly detection and automatic labeling for solar cell quality inspection based on generative adversarial network. *Sensors* 21(13), DOI 10.3390/s21134361, URL https://www.mdpi.com/1424-8220/21/13/4361

[11] Bartler A, Mauch L, Yang B, Reuter M, Stoicescu L (2018) Automated detection of solar cell defects with deep learning. In: 26th European Signal Processing Conference, EUSIPCO 2018, Roma, Italy, September 3-7, 2018, IEEE, pp 2035–2039, DOI 10.23919/EUSIPCO.2018.8553025

[12] Baur C, Wiestler B, Albarqouni S, Navab N (2018) Deep autoencoding models for unsupervised anomaly segmentation in brain mr images. In: International MICCAI Brainlesion Workshop, Springer, pp 161–169

[13] Bergmann P, Batzner K, Fauser M, Sattlegger D, Steger C (2021) The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision* pp 1–22

[14] Bergmann P, Batzner K, Fauser M, Sattlegger D, Steger C (2022) Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *Int J Comput Vis* 130(4):947–969, DOI 10.1007/s11263-022-01578-9

[15] Bin L, Xianghao H, Shuai F (2011) Automatic inspection of surface crack in solar cell images. In: 2011 Chinese Control and Decision Conference (CCDC), IEEE, pp 993–998

[16] Cauchy A, et al. (1847) Méthode générale pour la résolution des systemes d'équations simultanées. *Comp Rend Sci Paris* 25(1847):536–538

[17] Centre/BNEF FSU (2019) global trends in renewable energy. URL https://www.fs-unep-centre.org/global-trends-in-renewable-energy-investment-2019/

[18] Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41(3):1–58

[19] Chawla R, Singal P, Garg AK (2018) A mamdani fuzzy logic system to enhance solar cell micro-cracks image processing. *3D Research* 9(3):34

[20] Chen A, Zhou T, Icke I, Parimal S, Dogdas B, Forbes J, Sampath S, Bagchi A, Chin C (2017) Transfer learning for the fully automatic segmentation of left ventricle myocardium in porcine cardiac cine MR images. In: Pop M, Sermesant M, Jodoin P, Lalande A, Zhuang X, Yang G, Young AA, Bernard O (eds) Statistical Atlases and Computational Models of the Heart. ACDC and MMWHS Challenges - 8th International Workshop, STACOM 2017, Held in Conjunction with MICCAI 2017, Quebec City, Canada, September 10-14, 2017, Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 10663, pp 21–31, DOI 10.1007/978-3-319-75541-0_3, URL https://doi.org/10.1007/978-3-319-75541-0_3

[21] Chen H, Pang Y, Hu Q, Liu K (2018) Solar cell surface defect inspection based on multispectral convolutional neural network. *Journal of Intelligent Manufacturing* DOI 10.1007/s10845-018-1458-z, URL https://doi.org/10.1007/s10845-018-1458-z

[22] Chen H, Zhao H, Han D, Yan H, Zhang X, Liu K (2018) Robust crack defect detection in inhomogeneously textured surface of near infrared images. In: Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Springer, pp 511–523

[23] Chen H, Liu J, Wang S, Liu K (2019) Robust dislocation defects region segmentation for polysilicon wafer image with random texture background. *IEEE Access* 7:134318–134329

[24] Chen H, Zhao H, Han D, Liu K (2019) Accurate and robust crack detection using steerable evidence filtering in electroluminescence images of solar cells. *Optics and Lasers in Engineering* 118:22–33

[25] Chen H, Zhao H, Han D, Liu W, Chen P, Liu K (2019) Structure aware based crack defect detection for multicrystalline solar cells. *Measurement* p 107170

[26] Chen H, Hu Q, Zhai B, Chen H, Liu K (2020) A robust weakly supervised learning of deep conv-nets for surface defect inspection. *Neural Computing and Applications* pp 1–16

[27] Chen L, Yang Q, Yan W (2019) Generative adversarial network based data augmentation for pv module defect pattern analysis. In: 2019 Chinese Control Conference (CCC), pp 8422–8427, DOI 10.23919/ChiCC.2019.8866155

[28] Chintala S, Denton E, Arjovsky M, Mathieu M (2016) How to train a gan? tips and tricks to make gans work. URL https://github.com/soumith/ganhacks

[29] Chollet F, et al. (2018) *Deep learning with Python*, vol 361. Manning New York

[30] Crum WR, Camara O, Hill DL (2006) Generalized overlap measures for evaluation and validation in medical image analysis. *IEEE transactions on medical imaging* 25(11):1451–1461

[31] Csurka G, Larlus D, Perronnin F, Meylan F (2013) What is a good evaluation measure for semantic segmentation?. In: BMVC, Citeseer, vol 27, p 2013

[32] Deitsch S, Christlein V, Berger S, Buerhop-Lutz C, Maier A, Gallwitz F, Riess C, 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, DOI 10.1016/j.solener.2019.02.067

[33] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09

[34] Deng Y, Guo X, Wei Y, Lu K, Fang B, Guo D, Liu H, Sun F (2019) Deep reinforcement learning for robotic pushing and picking in cluttered environment. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 619–626, DOI 10.1109/IROS40897.2019.8967899

[35] Dong N, Xing EP (2018) Few-shot semantic segmentation with prototype learning. BMVC

[36] Dotenco S, Dalsass M, Winkler L, Würzner T, Brabec C, Maier A, Gallwitz F (2016) Automatic detection and analysis of photovoltaic modules in aerial infrared imagery. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, pp 1–9

[37] Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research* 12(7)

[38] Ferguson MK, Ronay A, Lee YTT, Law KH (2018) Detection and segmentation of manufacturing defects with convolutional neural networks and transfer learning. *Smart and sustainable manufacturing systems* 2:10.1520/SSMS20180033, DOI 10.1520/SSMS20180033, URL https://pubmed.ncbi.nlm.nih.gov/31093604

[39] Fernandez-Moral E, Martins R, Wolf D, Rives P (2018) A new metric for evaluating semantic segmentation: leveraging global and contour accuracy. In: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, pp 1051–1056

[40] Fuyuki T, Kitiyanan A (2009) Photographic diagnosis of crystalline silicon solar cells utilizing electroluminescence. *Applied Physics A* 96(1):189–196

[41] Fuyuki T, Kondo H, Yamazaki T, Takahashi Y, Uraoka Y (2005) Photographic surveying of minority carrier diffusion length in polycrystalline silicon solar cells by electroluminescence. 86:262108, DOI 10.1063/1.1978979

[42] Gao X, Munson E, Abousleman GP, Si J (2015) Automatic solar panel recognition and defect detection using infrared imaging. In: Automatic Target Recognition XXV, International Society for Optics and Photonics, vol 9476, p 94760O

[43] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial nets. In: Advances in neural information processing systems, pp 2672–2680

[44] Goodfellow I, Bengio Y, Courville A (2016) *Deep Learning*. MIT Press

[45] Gulrajani I, Ahmed F, Arjovsky M, Dumoulin V, Courville A (2017) Improved training of wasserstein gans. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, NIPS'17, p 5769–5779

[46] Haselmann M, Gruber DP, Tabatabai P (2018) Anomaly detection using deep learning based image completion. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), pp 1237–1242, DOI 10.1109/ICMLA.2018.00201

[47] He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

[48] He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969

[49] Hinton G, Srivastava N, Swersky K (2012) Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[50] IEA (2021) Renewables 2021. URL https://www.iea.org/news/global-energy-investments-set-to-recover-in-2021-but-remain-far-from-a-net-zero-pathway

[51] Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, PMLR, pp 448–456

[52] ISE F (2019) Photovoltaics report

[53] Jiang Y, Zhao C, Ding W, Hong L, Shen Q (2020) Attention m-net for automatic pixel-level micro-crack detection of photovoltaic module cells in electroluminescence images. In: 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS), IEEE, pp 1415–1421

[54] Kignma DP, Ba J (2014) Adam: A method for stochastic optimization. *Arxiv preprint arxiv:14126980*

[55] Ko J, Rheem J (2010) Anisotropic diffusion based micro-crack inspection in poly-crystalline solar wafers. In: World Congress on Engineering 2012. July 4-6, 2012. London, UK., International Association of Engineers, vol 2188, pp 524–528

[56] Ko J, Rheem J (2016) Defect detection of polycrystalline solar wafers using local binary mean. *The International Journal of Advanced Manufacturing Technology* 82(9):1753–1764, DOI 10.1007/s00170-015-7498-z, URL https://doi.org/10.1007/s00170-015-7498-z

[57] Ko SS, Liu CS, Lin YC (2013) Optical inspection system with tunable exposure unit for micro-crack detection in solar wafers. *Optik* 124(19):4030 – 4035, DOI https://doi.org/10.1016/j.ijleo.2012.12.024, URL http://www.sciencedirect.com/science/article/pii/S0030402613000260

[58] Kohli P, Torr PH, et al. (2009) Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision* 82(3):302–324

[59] Köntges M, Kunze I, Kajari-Schröder S, Breitenmoser X, Bjørneklett B (2011) The risk of power loss in crystalline silicon based photovoltaic modules due to micro-cracks. *Solar Energy Materials and Solar Cells* 95(4):1131–1137

[60] Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25:1097–1105

[61] Kumar SS, Abraham DM, Jahanshahi MR, Iseley T, Starr J (2018) Automated defect classification in sewer closed circuit television inspections using deep convolutional neural networks. *Automation in Construction* 91:273–283, DOI https://doi.org/10.1016/j.autcon.2018.03.028, URL https://www.sciencedirect.com/science/article/pii/S0926580517309767

[62] Le Cun Y, Jackel LD, Boser B, Denker JS, Graf HP, Guyon I, Henderson D, Howard RE, Hubbard W (1989) Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine* 27(11):41–46

[63] Li WC, Tsai DM (2012) Wavelet-based defect detection in solar wafer images with inhomogeneous texture. *Pattern Recognition* 45(2):742 – 56, URL http://dx.doi.org/10.1016/j.patcog.2011.07.025

[64] Liu K, Han J, Chen H, Yan H, Yang P (2019) Defect detection on el images based on deep feature optimized by metric learning for imbalanced data. In: 2019 25th International Conference on Automation and Computing (ICAC), IEEE, pp 1–5

[65] Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440

[66] Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440

[67] Luo Z, Cheng S, Zheng Q (2019) Gan-based augmentation for improving cnn performance of classification of defective photovoltaic module cells in electroluminescence images. In: IOP Conference Series: Earth and Environmental Science, IOP Publishing, vol 354, p 012106

[68] Maestro-Watson D, Balzategui J, Eciolaza L, Arana-Arexolaleiba N (2018) Deep learning for deflectometric inspection of specular surfaces. In: The 13th International Conference on Soft Computing Models in Industrial and Environmental Applications, Springer, pp 280–289

[69] Maestro-Watson D, Balzategui J, Eciolaza L, Arana-Arexolaleiba N (2019) Deflectometric data segmentation based on fully convolutional neural networks. In: Fourteenth International Conference on Quality Control by Artificial Vision, International Society for Optics and Photonics, vol 11172, p 1117209

[70] Maestro-Watson D, Balzategui J, Eciolaza L, Arana-Arexolaleiba N (2020) Deflectometric data segmentation for surface inspection: a fully convolutional neural network approach. *Journal of Electronic Imaging* 29(4):041007

[71] Mayr M, Hoffmann M, Maier A, Christlein V (2019) Weakly supervised segmentation of cracks on solar cells using normalized lp norm. In: 2019 IEEE International Conference on Image Processing (ICIP), pp 1885–1889, DOI 10.1109/ICIP.2019.8803116

[72] Movshovitz-Attias Y, Toshev A, Leung TK, Ioffe S, Singh S (2017) No fuss distance metric learning using proxies. In: Proceedings of the IEEE International Conference on Computer Vision, pp 360–368

[73] Otamendi U, Martinez I, Quartulli M, Olaizola IG, Viles E, Cambarau W (2021) Segmentation of cell-level anomalies in electroluminescence images of photovoltaic modules. *Solar Energy* 220:914–926, DOI https://doi.org/10.1016/j.solener.2021.03.058, URL https://www.sciencedirect.com/science/article/pii/S0038092X21002462

[74] Pardamean B, Cenggoro TW, Rahutomo R, Budiarto A, Karuppiah EK (2018) Transfer learning from chest x-ray pre-trained convolutional neural network for learning mammogram data. *Procedia Computer Science* 135:400–407, DOI https://doi.org/10.1016/j.procs.2018.08.190, URL https://www.sciencedirect.com/science/article/pii/S1877050918314807

[75] Qi H, Brown M, Lowe DG (2018) Low-shot learning with imprinted weights. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5822–5830

[76] Qian X, Li J, Cao J, Wu Y, Wang W (2020) Micro-cracks detection of solar cells surface via combing short-term and long-term deep features. *Neural Networks*

[77] Qian X, Li J, Zhang J, Zhang W, Yue W, Wu QE, Zhang H, Wu Y, Wang W (2020) Micro-crack detection of solar cell based on adaptive deep features and visual saliency. *Sensor Review*

[78] Rahman MRU, Chen H (2020) Defects inspection in polycrystalline solar cells electroluminescence images using deep learning. *IEEE Access* 8:40547–40558, DOI 10.1109/ACCESS.2020.2976843

[79] Rakelly K, Shelhamer E, Darrell T, Efros A, Levine S (2018) Conditional networks for few-shot semantic segmentation.

[80] Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788

[81] Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, pp 234–241

[82] Ruder S (2016) An overview of gradient descent optimization algorithms. *CoRR* abs/1609.04747, URL http://arxiv.org/abs/1609.04747, 1609.04747

[83] Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. *nature* 323(6088):533–536

[84] Schlegl T, Seeböck P, Waldstein SM, Langs G, Schmidt-Erfurth U (2019) f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical image analysis* 54:30–44

[85] Shaban A, Bansal S, Liu Z, Essa I, Boots B (2017) One-shot learning for semantic segmentation. URL http://dblp.uni-trier.de/db/conf/bmvc/bmvc2017.html#ShabanBLEB17

[86] Siam M, Oreshkin B, Jagersand M (2019) Amp: Adaptive masked proxies for few-shot segmentation. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp 5248–5257, DOI 10.1109/ICCV.2019.00535

[87] Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:14091556*

[88] Spataru S, Hacke P, Sera D (2016) Automatic detection and evaluation of solar cell micro-cracks in electroluminescence images using matched filters. In: 2016 IEEE 43rd Photovoltaic Specialists Conference (PVSC), pp 1602–1607, DOI 10.1109/PVSC.2016.7749891

[89] Staar B, Lütjen M, Freitag M (2019) Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP* 79:484–489

[90] Stromer D, Vetter A, Oezkan HC, Probst C, Maier A (2019) Enhanced crack segmentation (ecs): A reference algorithm for segmenting cracks in multicrystalline silicon solar cells. *IEEE Journal of Photovoltaics* 9(3):752–758

[91] Su B, Chen H, Zhu Y, Liu W, Liu K (2019) Classification of manufacturing defects in multicrystalline solar cells with novel feature descriptor. *IEEE Transactions on Instrumentation and Measurement* 68(12):4675–4688

[92] Sun M, Lv S, Zhao X, Li R, Zhang W, Zhang X (2017) Defect detection of photovoltaic modules based on convolutional neural network. In: International Conference on Machine Learning and Intelligent Communications, Springer, pp 122–132

[93] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9

[94] Tsai D, Luo J (2011) Mean shift-based defect detection in multicrystalline solar wafer surfaces. *IEEE Transactions on Industrial Informatics* 7(1):125–135, DOI 10.1109/TII.2010.2092783

[95] 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, DOI 10.1109/TII.2012.2209663

[96] Tsai D, Fan MSK, Huang Y, Chiu W (2019) Saw-mark defect detection in heterogeneous solar wafer images using gan-based training samples generation and cnn classification. In: Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,, INSTICC, SciTePress, pp 234–240, DOI 10.5220/0007306602340240

[97] Tsai DM, Chang CC, Chao SM (2010) Micro-crack inspection in heterogeneously textured solar wafers using anisotropic diffusion. *Image and Vision Computing* 28(3):491–501

[98] Tsai DM, Wu SC, Li WC (2012) Defect detection of solar cells in electroluminescence images using fourier image reconstruction. *Solar Energy Materials and Solar Cells* 99:250–262

[99] Tsai DM, Li GN, Li WC, Chiu WY (2015) Defect detection in multi-crystal solar cells using clustering with uniformity measures. *Advanced Engineering Informatics* 29(3):419–430

[100] Tsai YH, Hung WC, Schulter S, Sohn K, Yang MH, Chandraker M (2018) Learning to adapt structured output space for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 7472–7481

[101] Tseng DC, Liu YS, Chou CM (2015) Automatic finger interruption detection in electroluminescence images of multicrystalline solar cells. *Mathematical Problems in Engineering* 2015

[102] University S (2016) Cs231n convolutional neural networks for visual recognition. URL https://cs231n.github.io/convolutional-networks/

[103] Vaněk J, Repko I, Klima J (2016) Automation capabilities of solar modules defect detection by thermography. *ECS Transactions* 74(1):293–303

[104] Wang M, Deng W (2018) Deep visual domain adaptation: A survey. *Neurocomputing* 312:135–153, DOI https://doi.org/10.1016/j.neucom.2018.05.083, URL https://www.sciencedirect.com/science/article/pii/S0925231218306684

[105] Wang T, Chen Y, Qiao M, Snoussi H (2018) A fast and robust convolutional neural network-based defect detection model in product quality control. *The International Journal of Advanced Manufacturing Technology* 94(9-12):3465–3471

[106] Wang X, Barnett A (2019) The evolving value of photovoltaic module efficiency. *Applied Sciences* 9(6):1227

[107] Wang X, Li J, Yao M, He W, Qian Y (2014) Solar cells surface defects detection based on deep learning. *Pattern Recognition & Artificial Intelligence* 27(6):517–523

[108] Wang Y, Liu M, Zheng P, Yang H, Zou J (2020) A smart surface inspection system using faster r-cnn in cloud-edge computing environment. *Advanced Engineering Informatics* 43:101037, DOI https://doi.org/10.1016/j.aei.2020.101037, URL https://www.sciencedirect.com/science/article/pii/S1474034620300069

[109] Wu W, Li H, Li X, Guo H, Zhang L (2019) Polsar image semantic segmentation based on deep transfer learning–realizing smooth classification with small training sets. *IEEE Geoscience and Remote Sensing Letters*

[110] Xu P, Zhou W, Fei M (2014) Detection methods for micro-cracked defects of photovoltaic modules based on machine vision. In: 2014 IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, IEEE, pp 609–613

[111] Yang Y, Meng F, Li H, Ngan KN, Wu Q (2019) A new few-shot segmentation network based on class representation.

[112] Yen H, Sie Y (2012) Machine vision system for surface defect inspection of printed silicon solar cells. In: The 1st IEEE Global Conference on Consumer Electronics 2012, pp 422–424, DOI 10.1109/GCCE.2012.6379645

[113] Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? *CoRR* abs/1411.1792, URL http://arxiv.org/abs/1411.1792, 1411.1792

[114] Zeng A, Song S, Welker S, Lee J, Rodriguez A, Funkhouser T (2018) Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 4238–4245, DOI 10.1109/IROS.2018.8593986

[115] Zhang C, Lin G, Liu F, Yao R, Shen C (2019) Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. *arXiv preprint arXiv:190302351*

[116] Zhang X, Sun H, Zhou Y, Xi J, Li M (2013) A novel method for surface defect detection of photovoltaic module based on independent component analysis. *Mathematical Problems in Engineering* 2013

[117] Zhang X, Wei Y, Yang Y, Huang T (2018) Sg-one: Similarity guidance network for one-shot semantic segmentation. 1810.09091v3

[118] Zhang X, Hao Y, Shangguan H, Zhang P, Wang A (2020) Detection of surface defects on solar cells by fusing multi-channel convolution neural networks. *Infrared Physics & Technology* p 103334

[119] Zhi W, Yueng HWF, Chen Z, Zandavi SM, Lu Z, Chung YY (2017) Using transfer learning with convolutional neural networks to diagnose breast cancer from histopathological images. In: Liu D, Xie S, Li Y, Zhao D, El-Alfy ESM (eds) Neural Information Processing, Springer International Publishing, Cham, pp 669–676

[120] Zhuang F, Yanzheng Z, Yang L, Qixin C, Mingbo C, Jun Z, Lee J (2004) Solar cell crack inspection by image processing. In: Proceedings of 2004 International Conference on the Business of Electronic Product Reliability and Liability (IEEE Cat. No. 04EX809), IEEE, pp 77–80