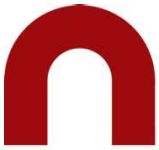


Instituto Superior de Engenharia

Politécnico de Coimbra

MESTRADO EM ENGENHARIA ELETROTÉCNICA

Automated Visual Inspection of Electrical Grid Assets using Deep Learning

Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

Pedro Daniel Carvalheiro Rocha

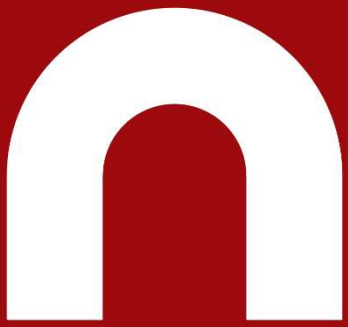
Orientador

Professor Doutor Fernando José Pimentel Lopes

Supervisor na empresa EDP LabElec

Eng. André Coelho

Coimbra, Março 2024



INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Resumo

A crescente procura por energia elétrica, estimada para duplicar até 2050, obriga a atualizações nos sistemas de Produção de Energia e Transporte e Distribuição (T&D). Para atender às necessidades, são necessárias novas infraestruturas. No entanto, a fiabilidade dos sistemas de produção e distribuição elétrica em serviço é bem mais critica.

Por um lado, uma das direções seguidas para a produção de energia é o aumento do uso de tecnologias fotovoltaicas (PV), com a capacidade global de energia solar a ultrapassar 1 TW em 2022. Com a expansão dos sistemas PV, a inspeção manual de ativos torna-se impraticável, levando à utilização de *drones* para a recolha remota de imagens. Por outro lado, nas próximas três décadas, os sistemas de T&D podem exceder 160 milhões de km mundialmente, onde os isoladores de AT representam mais de 50% dos custos de manutenção. As empresas do sector estão a colecionar quantidades insustentáveis de imagens como parte do processo de inspeção de infraestruturas elétricas.

Tanto na produção de energia PV quanto nos sistemas de T&D, o aumento do volume de dados adquiridos e o seu processamento está atualmente a ser estrangulado por tarefas de interpretação humana, dado que aumentar a escala desse processo através de formação é muito demorado. No entanto, a maioria das tarefas de inspeção com responsabilidade reduzida pode ser automatizada através da utilização de tecnologia.

Este trabalho demonstra o potencial da visão computacional e aprendizagem profunda para inspeção visual automatizada, validada por dados reais, expondo vantagens e limitações. O projeto revê avanços recentes na inspeção de ativos elétricos, nomeadamente painéis fotovoltaicos e isoladores, e condensa o conhecimento essencial em inspeção de ativos, visão computacional e inteligência artificial. Apresenta ainda um conjunto de ferramentas atualmente usadas para desenvolver projetos de aprendizagem profunda.

O trabalho desenvolvido inclui exemplos de aplicações de inspeção visual automatizada de Células PV, Módulos PV e Isoladores de AT. A condição operacional de Células PV é classificada ao processar imagens EL com a rede VGG19, obtendo os valores de AUC de 0,95, 0,94 e 0,90 para os *datasets* poli, mono e misto, respetivamente, apresentando melhorias globais em comparação com as reportadas na literatura. Imagens IR de Módulos PV são analisadas usando quatro CNNs personalizadas com diferentes níveis de complexidade para classificar padrões termográficos em classes defeituosas pré-definidas, alcançando valores de Precisão, Recall e F1-score de 0,87, 0,86 e 0,86, respectivamente. Defeitos em isoladores de AT são detetados por dois modelos de ultima geração com recurso a imagens de luz visível. O YOLOv8s alcança uma mAP@50 de 87,9%, enquanto o Faster R-CNN X101-FPN alcança 87,2% para a mesma métrica.

Este trabalho pode promover o desenvolvimento de um modelo robusto que permita que empresas do sector beneficiem de processos de inspeção mais eficientes.

Palavras-chave: Visão, Aprendizagem Profunda, Inspeção de Ativos, Fotovoltaico, Isolador.

Abstract

The global growing demand for electric power, projected to double by 2050, requires extensive upgrades in both Power Generation and Transport and Distribution (T&D) systems. To address these needs, new infrastructures are required. Meanwhile, the reliability of the aging electrical generation and distribution systems is critical.

On the one hand, one main direction for power generation is the increased use of Photovoltaic (PV) technologies., with the global solar power capacity exceeding 1 TW in 2022. With the rapid expansion of PV systems, manual image collection for asset inspection becomes increasingly impractical, leading to imagery collected by UAVs. On the other hand, in the next three decades, worldwide T&D systems may exceed 160 million km, where High Voltage (HV) power line insulators account for over 50% of the maintenance costs. Electric utility companies are increasingly collecting visual data as part of their inspection process of electrical T&D infrastructures.

In both PV power generation and T&D systems, the increasing volume of collected data and its processing, is currently being limited by human interpretation tasks, mainly because escalating this pipeline element trough training is very expensive. However, most inspection tasks with minor responsibility can be automated using computer vision and deep learning techniques to support the system's growing demand.

This work demonstrates the potential of computer vision and deep learning for automated visual inspection of electrical grid assets, validated with real-world data, to reveal advantages and limitations. The project reviews recent advancements in automated visual inspection of electrical assets, namely PV panels and HV insulators, and outlines the essential knowledge in asset inspection, computer vision and deep learning. It also introduces a set of tools currently used to deploy deep learning projects.

The developed work includes example applications on automated visual inspection of PV Cells, PV Modules and HV Insulators. PV Cells' operational condition is classified by processing EL images using the VGG19, resulting in AUC values of 0.95, 0.94, and 0.90 for the poly, mono, and mixed datasets, respectively, with overall improvements compared to those reported in the literature. IR images of PV Modules are analyzed using four custom shallow CNNs with varying levels of complexity to classify thermographic patterns into predefined defective classes, achieving Precision, Recall, and F1-score values of 0.87, 0.86, and 0.86, respectively. Defects in HV insulators are detected using two SOTA deep learning models with visible light images. YOLOv8s achieves a mAP@50 of 87.9% while Faster R-CNN X101-FPN achieves 87.2% for the same metric.

This work can serve as an encouragement in developing a robust model, allowing utility companies to benefit from higher efficiency inspection processes.

Keywords: Computer Vision, Deep Learning, Asset Inspection, Photovoltaic, HV Insulator.

Epigraph

"Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination."

attributed to Albert Einstein

Dedication

I dedicate this dissertation to my loving family.

To my mother, whose unshakable support, faith, and sacrifices have been the bedrock of my journey. And to my siblings, who have always believed in me and my dreams.

Acknowledgements

Throughout this year, many individuals played crucial roles in my life. Some have significantly contributed to my growth as an engineer, while others provided unwavering support during challenging times.

My sincere appreciation to my supervisor, Professor Fernando Lopes, for his enlightening guidance and consistent support throughout my research. His attendance and readiness have been invaluable.

I am deeply thankful to Researcher Nuno Martins of IT-Coimbra for his invaluable tips, suggestions, and for kickstarting this learning journey.

A special thanks to EDP LabElec, Eng. André Coelho, Benjamin Correia, Ricardo Santos and João Santos for their insights and collaboration in validating our study in real-data applications..

I am grateful to all the professors of ISEC DEE for their interactive knowledge sharing.

And I also want to express my gratitude to my classmates for inspiring my journey and for resonating with my own struggles.

Additionally, I would like to express my gratitude to my family and friends for their unwavering encouragement and moral support throughout my studies.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BackProp	Backpropagation
Conv2D	Convolutional layer of 2 Dimensions
ConvNet	Convolutional Neural Network
Grad-CAM	Gradient-weighted Class Activation Mapping
Id	Identifier
LiDAR	Light Detection And Ranging
Mono	Monocrystalline
Poly	Polycrystalline
RaDAR	Radio Detection And Ranging
ReLU	Rectified Linear Unit (activation function)
ResNet	Residual Network
SoNAR	Sound Navigation And Ranging
Tech	Technology

List of Acronyms

AI	Artificial Intelligence
AM	Asset Management
API	Application Programming Interface
AR	Augmented Reality
AUC	Area Under the Curve
AVIA	Automated Visual Inspection of Assets
BB	Bounding Box
BGR	Blue Green Red
CNN	Convolutional Neural Network
CORDIC	COordinate Rotation DIgital Compute
CPU	Central Processing Unit
CV	Computer Vision
DL	Deep Learning
DSO	Distribution System Operator
DV	Deep Vision
EL	Electro-Luminescence
EPRI	Electric Power Research Institute
FAIR	Facebook AI Research
FF	Fill Factor
GAN	Generative Adversarial Networks
GD	Gradient Descent
GIS	Geographic Information Systems
GPU	Graphics Processing Unit
GT	Ground Truth
GUI	Graphical User Interface
HV	High Voltage
IDE	Integrated Development Environment
IDID	Insulator Defect Image Dataset
IEEE	Institute of Electrical and Electronics Engineers
IoU	Intersection over Union
IR	Infra-Red
ISEC	Instituto Superior de Engenharia de Coimbra
JSON	Java-Script Object Notation
LIME	Local Interpretable Model-agnostic Explanations
mAP	mean Average Precision
mAR	mean Average Recall
MFNN	Modified Fuzz Neural Network

ML	Machine Learning
MP	Mega-Pixels
MS-COCO	Microsoft Common Objects in Context
MV	Medium Voltage
MTTF	Mean Time To Failure
NDT	Non-Destructive Testing
NMS	Non-Maximum Suppression
O&M	Operation and Maintenance
OCR	Optical Character Recognition
OC	Open Circuit
PDF	Portable Document Format
PID	Potential Induced Degradation
PV	Photo-Voltaic
QC	Quality Control
R&D	Research and Development
R-CNN	Region-based Convolutional Neural Network
RCNN	Residual Convolutional Neural Network
RF	Random Forest
RGB	Red Green Blue
ROC	Receiver Operating Characteristic
ROI	Region Of Interest
RPN	Region Proposal Network
SC	Short Circuit
SHAP	SHapley Additive exPlanations
SM	Smart Management
SMOTE	Synthetic Minority Over-sampling TEchnique
SOTA	State-Of-The-Art
SSD	Single Shot Detector
SVM	Support Vector Machine
STC	Standard Test Conditions
T&D	Transmission and Distribution
TPU	Tensor Processing Unit
UAV	Unmanned Aerial Vehicle
UV	Ultra-Violet
VAE	Variational Auto-Encoders
YOLO	You Only Look Once

List of Symbols

α	Learning rate
a	Activation (e.g., output of a neuron or layer)
b	Bias vector
D	Number of features or dimensions
E	Epoch (a complete pass through the training dataset)
f	Neural network model (function)
$\mathcal{L}$	Loss or cost function
L	Layer of a neural network
M	Mini-batch size
N	Number of data samples
km	Kilometer
kW	Kilowatt
kWh	Kilowatt-hour
σ	Sigmoid (Logistic) activation function
$\tanh$	Hyperbolic tangent activation function
TW	Terawatt
TWh	Terawatt-hour
θ	Model parameters (e.g., weights and biases)
W	Weight matrix
X	Input data or feature matrix
Y	Output or target variable
y	True or actual output
$\hat{y}$	Predicted output
z	Linear combination of weights and inputs

1 Introduction

As an electrical engineering master's student, the primary goal of this project is to demonstrate a solid understanding of applied computer vision in the domain of electrical engineering, validating the assimilation of knowledge, theory and methodologies acquired throughout the degree program.

Additionally, this work attempts to improve communication skills, develop critical thinking and research capabilities, as well as foster time management.

Finally, a global objective of this work is to contribute with novel insights or perspectives on the existing body of knowledge in the field of electrical engineering.

This chapter presents an overview of the document's organization, emphasizing the developed work and the obtained results, while highlighting the relevance of this study for the thriving area of automated visual inspection of electrical grid asset solutions leveraged by deep learning.

1.1 Problem Statement

Electric power access is the structural backbone of modern society. Without it, it is challenging to imagine an outcome less than catastrophic. Global electricity demand is projected to double by 2050, exceeding 50,000 TWh annually, powered by 80% increase of wind and solar energy. Consequently, the worldwide Transmission and Distribution (T&D) grid expansion must follow this growth, reaching over 160 million km [1].

Developing new infrastructures to keep up with the escalating demand for electric power should be a concern. Yet, managing and maintaining existing electrical generation and distribution systems is critical, since the impact of their reliability is obviously greater. In the European Union, more than 50% of the grid has been in operation for over 20 years, which is approximately half of its average lifespan. To ensure that modern society can continue to depend on a stable and uninterrupted supply of electricity, regular maintenance, visual inspections, and predictive maintenance techniques are being employed to identify and address potential issues before they lead to failures.

In April 2022, the global solar power capacity reached 1 TW. Equivalent to 200 trillion square meters ($200 \cdot 10^6$ km^2), assuming a 20% efficiency of the 1000 watts striking each square meter of land. The annual Operation and Maintenance (O&M) cost of utility-scale, single-axis tracking solar systems is approximately $16.42 per kilowatt (kW)[2].

With this rapid expansion of modern photovoltaic systems, manual image collection and analysis become increasingly impractical. Instead, vast amounts of imagery are being collected by UAVs equipped with thermographic cameras, demanding for an automated analysis using computer vision and machine learning, that can be reliable.

Insulators account for only 5–8% of installation costs, yet they incur over 50% of maintenance expenses [3]. Electric utility companies are collecting visual data as part of their inspection process of electrical infrastructures, to detect potential flaws and to make informed decisions about maintenance prioritization.

Collecting aerial in-person inspection images costs range from 150 to 800 €/km, while automated aerial inspections cost between 40 and 150 €/km [4]. However, the increasing volume of collected data, especially images, during infrastructure inspections, together with its processing, has a bottle-neck in the human interpretation tasks, mainly because escalating this pipe-line element trough training is timely expensive. Yet, many aspects of the inspection process, particularly the initial sorting tasks with minor responsibilities, can be automated using computer vision and deep learning techniques to support the system's growing demand.

Advancing these technologies to a comprehensive and reliable level for widespread adoption and practical implementation through commercial applications requires further research and development efforts.

1.2 Objectives

The main objective of the work presented in this project report is to demonstrate the potential value of an automated system for visual inspections of electrical grid assets. The system should be based on computer vision techniques employing deep learning state-of-the-art models, as a viable hypothesis for adoption by electrical infrastructure maintenance managers. To achieve this objective, the following steps were followed:

- Reviewing the state-of-the-art for automated visual inspections of PV and electric grid assets, including technologies, methodologies, and challenges.
- Employing PV thermal imagery classification models to acquire intuition and practical implementation experience, establishing a foundational understanding before progressing to more complex models.
- Developing an object detection model to assess the condition of HV ceramic insulators, comparing different advanced deep learning architecture trade-offs.
- Establishing connections with major stakeholders to gather real-world data for inference and evaluation of the attained performance.
- Planning potential further development directions for improved work on the electrical grid asset management field.

1.3 Key Contributions

The project main contributions are the practical insights across several key areas that can benefit those interested in pursuing similar work, through these specific points:

- A succinct yet in-depth theoretical explanation of basic concepts that can provide a shortcut for newcomers in the field to grasp the essentials for further study.
- A condensed and structured hands-on guide to setting up deep learning projects, featuring valuable tips and many cross-references.
- Two example projects with practical insights for binary classification of PV Cells' operational conditions, and for multi-class classification of defective PV Modules.
- A comparative analysis of two State-Of-The-Art (SOTA) object detectors using high resolution images of High-Voltage (HV) Insulators, assessing their strengths, weaknesses, and performance differences.
- A potential demonstration for industry professionals seeking to implement automated analysis techniques, with the goal of enhancing the maintenance efficiency of electrical Transmission and Distribution (T&D) systems.

1.4 Document Overview

- **Chapter 1 - Introduction:** Provides a layman overview of the document's scope, objectives, contributions, and the roadmap for subsequent chapters.
- **Chapter 2 - Theoretical Background:** Presents the core concepts of Asset Management, Computer Vision and Deep Learning, with enough latitude to allow individuals in engineering fields to gain intuition to comprehend and critically assess the forthcoming work on Automated Visual Inspection of Assets (AVIA).
- **Chapter 3 - Framework and Design Methodologies:** Briefly explores local and cloud-based platforms that use the Jupyter Notebook software environment, and introduces essential libraries/APIs. It presents examples of coding notebooks, providing guidelines and references for replicating the project outcomes.
- **Chapter 4 - Case Studies and Implementations:** Presents and discusses three implemented projects: PV Cells' operational condition classification by processing Electro-Luminescence (EL) images with a classical CNN; PV Modules' Infra-Red (IR) image defects classification, achieved through a comparison of 4 custom CNNs with varying complexities; and High-Voltage (HV) ceramic insulators' defect detection using two different State-Off-The-Art (SOTA) models. The brand new version 8 of YOLO and the Faster R-CNN with a ResNetX backbone.
- **Chapter 5 - Conclusion:** Summarizes findings and contributions and proposes future research paths, setting expectations for ongoing advancements in the field.

2 Theoretical Background

This chapter is structured with the aim that anyone in the field of engineering will develop the intuition necessary to confidently understand and critically assess the work presented in the subsequent chapters.

It establishes an irregular path, compiling converging routes, beginning by elucidating the core concepts of **Asset Management** (**AM**), continuing by exploring the recent practical developments of **Smart Management** (**SM**) and narrowing down to the maintenance perspective of conducting **Quality Control** (**QC**) visual inspections. This path is leveraged by unfolding the **Computer Vision** (**CV**) scope and a detailed presentation of the **Deep Learning** (**DL**) theory, from the foundational notions, to the state-of-the-art breakthroughs in **Deep Vision** (**DV**) applications.

This comprehensive presentation is justified by the assumption that the development of an **Automated Visual Inspection of Assets** (**AVIA**) system is supported by the convergent extension of these fields, also interpreted as their intersection in the proposed *Venn diagram* illustrated in Figure 2.1.

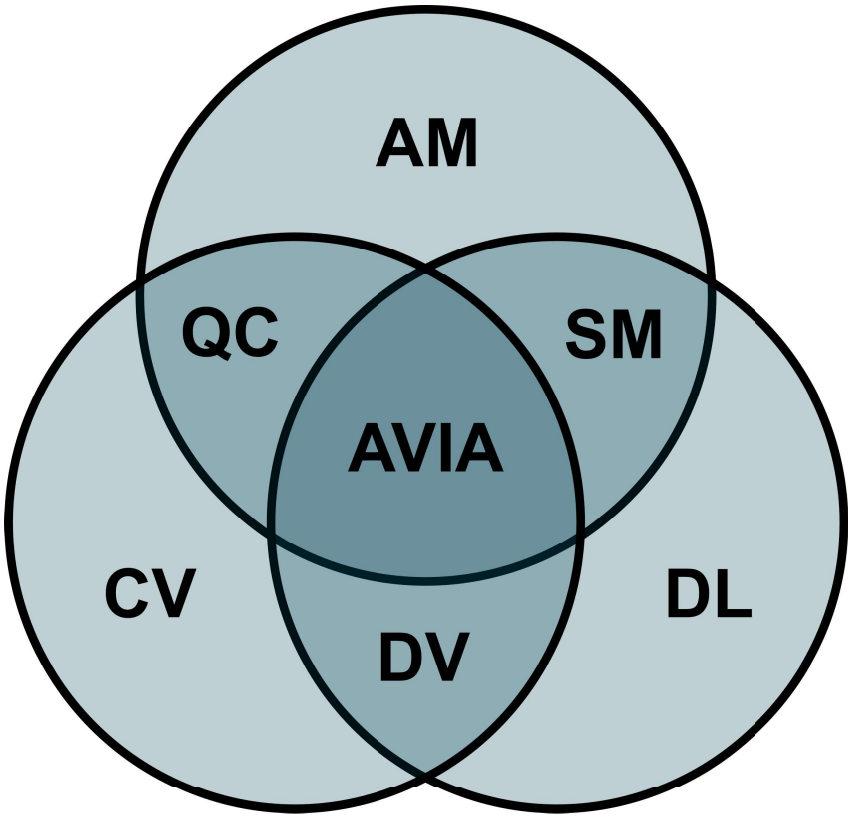


Figure 2.1: Proposed diagram for the fields of study involved in AVIA

2.1 Asset Management

Asset management optimization is of utmost importance for enhancing business performance and profitability, particularly within the electric power industry. Recent years have seen the emergence and adoption of the smart grid concept, making the planning, design, operation, and maintenance of expanding assets a central challenge.

Hence, there is a global demand for essential asset management tools and standards. The International Standards Organization (ISO), in response to this need, formally introduced the ISO 55000 [5].

The standard emphasizes the importance of aligning asset management with the strategic goals, integrated with the overall management systems, policies and procedures, to ensure that asset management activities employ consistency and efficiency, delivering value to the organization. Organizations should consider the needs and expectations of all relevant stakeholders, both internal and external, and cover the entire lifecycle of assets, from acquisition and design to operation, maintenance and disposal of assets. ISO 55000 emphasizes the importance of identifying, assessing, and managing risks associated with asset performance, safety, and compliance, and encourage the organizations to adopt a culture of continuous improvement in asset management, using data and performance indicators to drive optimization [6]. Main points:

- **Asset Management Software:** Many organizations use specialized software to plan, schedule, and track maintenance activities. These software systems can help with work order management, asset tracking, inventory control, and data analysis.
- **Lifecycle Management:** Asset maintenance is often part of a broader asset lifecycle management strategy, which includes asset acquisition, utilization, maintenance, and disposal. Effective management of an asset throughout its lifecycle is essential for achieving optimal ROI.
- **Key Performance Indicators (KPIs):** Organizations use KPIs to measure the effectiveness of their asset maintenance programs. Common KPIs include mean time between failures (MTBF), mean time to repair (MTTR), asset uptime, and maintenance cost as a percentage of the asset's value.
- **Total Cost of Ownership (TCO):** TCO is a financial concept used in asset maintenance that considers all the costs associated with an asset over its entire lifespan, including acquisition, maintenance, operating, and disposal costs.
- **Regulatory Compliance:** Some industries have specific regulations and standards governing asset maintenance, particularly in fields such as healthcare, aviation, and manufacturing. Compliance with these regulations is crucial to ensure safety and quality.

Considering the entire life cycle of assets is crucial for minimizing costs and being effective in asset management. By understanding and strategically managing each phase of an asset's life cycle, organizations can optimize performance, reduce unexpected failures, and make informed decisions about maintenance and replacement. Organizations can influence the characteristics of each phase through effective management practices:

- **Infant Mortality Phase:** Implementing robust quality control measures during design and manufacturing to reduce early failures by managing the investment in testing and commissioning to identify and address potential issues before assets enter operational use.
- **Constant Failure Rate Phase:** Implementing proactive and predictive maintenance strategies to address wear and tear, reducing the likelihood of random failures and utilizing condition monitoring and reliability-centered maintenance to identify and address potential issues before they escalate.
- **Wear-Out Phase:** Developing effective strategies for asset renewal or replacement based on predictive analytics and data-driven insights and taking into account considerations such as technological advancements, shifts in operational requirements, and the overall economic viability of assets.

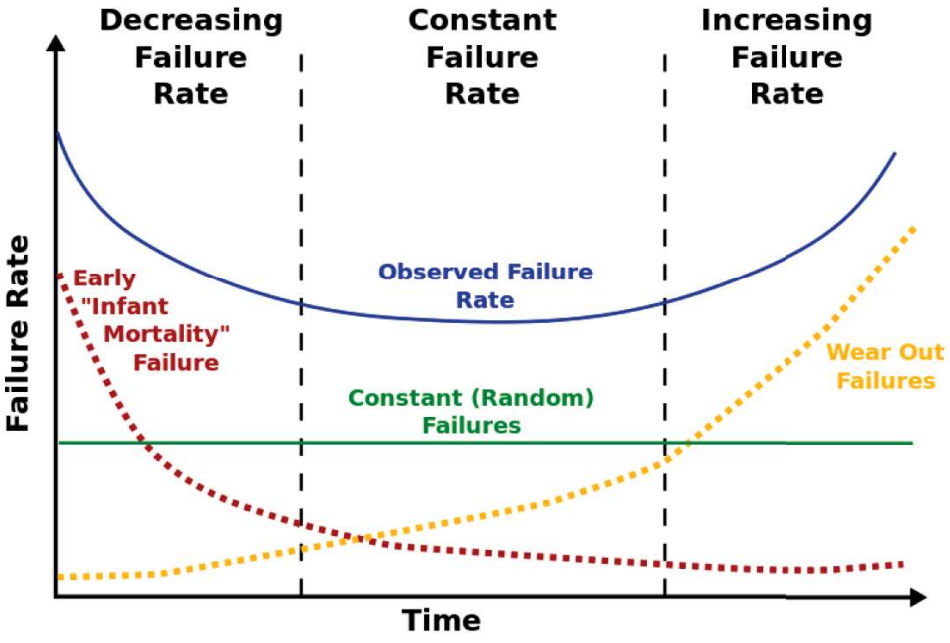


Figure 2.2: Failure rate over the lifecycle of a product [7]

By actively managing each phase of the life cycle and incorporating lessons learned from data analysis and experience, organizations can optimize the performance of their assets, extend their useful life, and make cost-effective decisions.

2.1.1 Maintenance Strategies and Practices

Effective asset maintenance requires a well-structured strategy, the use of modern technology and data analysis, and a commitment to continuous improvement. Many organizations use a combination of the following maintenance types, based on various criteria, such as the purpose, the timing, or the methods used.

- **Preventive Maintenance** involves regular inspections, servicing, and repair work, to prevent equipment or systems from breaking down. It aims to identify and address potential issues before they become major problems.
  - **Scheduled Maintenance** is performed at predefined intervals, regardless of the equipment's condition. It often includes tasks like routine inspections, lubrication, and calibration.
  - **Proactive Maintenance** overcome preventive measures by actively seeking ways to improve equipment reliability, implementing equipment redesign, upgrades, and process optimization to reduce maintenance requirements.
- **Corrective Maintenance** is performed in response to a failure or malfunction, often unplanned and reactive. It involves repairing or restoring equipment to its normal operating condition after a breakdown has occurred.
  - **Unscheduled Maintenance** is carried out as a response to unexpected failures, emergencies, or urgent repair needs. It is often costly and can result in downtime, but it is necessary to address critical issues promptly.
  - **Run-to-Failure Maintenance** is letting some equipment to run until they fail completely, particularly low-cost or non-critical assets. This approach is cost-effective for items that are inexpensive to replace and have minimal impact on operations when they fail.
- **Predictive Maintenance** uses data and predictive analytics to anticipate when equipment is likely to fail. By monitoring equipment health and performance indicators, maintenance can be scheduled just in time to prevent failures.
  - **Condition-Based Maintenance** is similar to predictive maintenance but focuses on the actual state and performance of equipment rather than fixed schedules, relying on real-time monitoring and data analysis of equipment conditions to make maintenance decisions.
  - **Reliability-Centered Maintenance** prioritizes critical assets and identifies the most cost-effective maintenance strategies for each asset, considering factors like safety, environmental impact, and operational goals.
  - **Total Productive Maintenance** aims to maximize the overall effectiveness of equipment and processes, by involving all employees in maintenance activities and focuses on reducing losses and improving productivity.

Utility companies employ a range of strategies to maintain the availability and reliability of electric power distribution systems [8]. Common practices include scheduled maintenance routines involving checks, servicing and replacements. These routines proactively identify and address deterioration before it leads to significant issues. Additionally, emergency response plans ensure prompt reactions to unexpected outages, swiftly mobilizing repair crews and efficiently allocating resources.

These tasks are facilitated by Digital Asset Management systems, that track equipment conditions and locations, maintenance history and performance data, prioritizing maintenance tasks and allocating resources effectively.

In recent decades, real-time remote monitoring of critical equipment has been implemented, enabling operators to identify anomalies without requiring a physical presence. Skilled technicians and engineers play a vital role in interpreting data, conducting thorough inspections, and making informed decisions based on their expertise. Critical equipment often utilizes sensors for real-time assessment of operational conditions. When deviations from normal parameters are detected, it triggers alerts for immediate investigation.

By combining sensor information and advanced data analysis, the aim is to predict equipment failures based on performance trends and operating conditions, enabling timely interventions, and subsequently reducing downtime and costs [9].

Sensors such as Infra-Red cameras are utilized to detect abnormal temperature variations in equipment, as demonstrated in Figure 2.3. Ultra-Violet inspection is employed to detect the ionization of the air surrounding a conductor, a phenomenon known as corona discharge, and in other cases, to identify contaminants on the surface of insulators. Ultrasound Technology is employed to detect sound waves emitted by malfunctioning equipment, revealing issues like electrical arcing, leaks, and mechanical faults. LIDAR can detect and analyze vegetation growth around power lines and substations.

Drones equipped with sensors such as cameras can inspect power lines, towers, insulators and substations from above, while ground-based robots access confined spaces and hazardous environments for inspections. AI and big data analytics can analyze extensive data to predict equipment failures by identifying patterns, enhancing maintenance decisions.

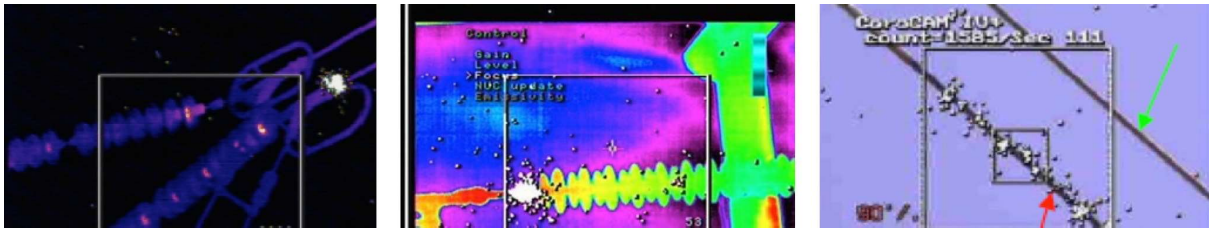


Figure 2.3: IR temperature and UV ionization anomalies detection [10]

2.1.1.1 Smart Management

Solar plant systems have intricate nonlinear dynamics that are susceptible to fluctuations and uncertainties in environmental conditions. Classical methods struggle to model these complexities effectively, while Machine Learning (ML) methods exhibit superior performance in handling such intricacies. Ekaterina Engel and Nikita Engel [11] extensively reviewed applications and the performance of machine learning techniques in PV power plants and presented a structured benchmark of recently published articles. As illustrated in Figure 2.4, typical applications include modeling and optimizing the design and size of solar plants, as well as forecasting the received irradiance and the generated power through regression. Additionally, reinforcement learning is employed to control the configuration of solar plants and optimize electronic energy conversion, also known as Maximum Power Point Tracking (MPPT). For Maintenance, they reviewed the imagery sensor techniques for failure diagnostics, and to conclude the study, they developed a fault forecasting system based on a Modified Fuzzy Neural Network (MFNN), that includes two RNNs with fuzzy units. That system effectively forecasted a failure-free operating period for a PV box with wiring losses with a relative error of 6E-4, tuned by a two-year historical dataset of PV signals.

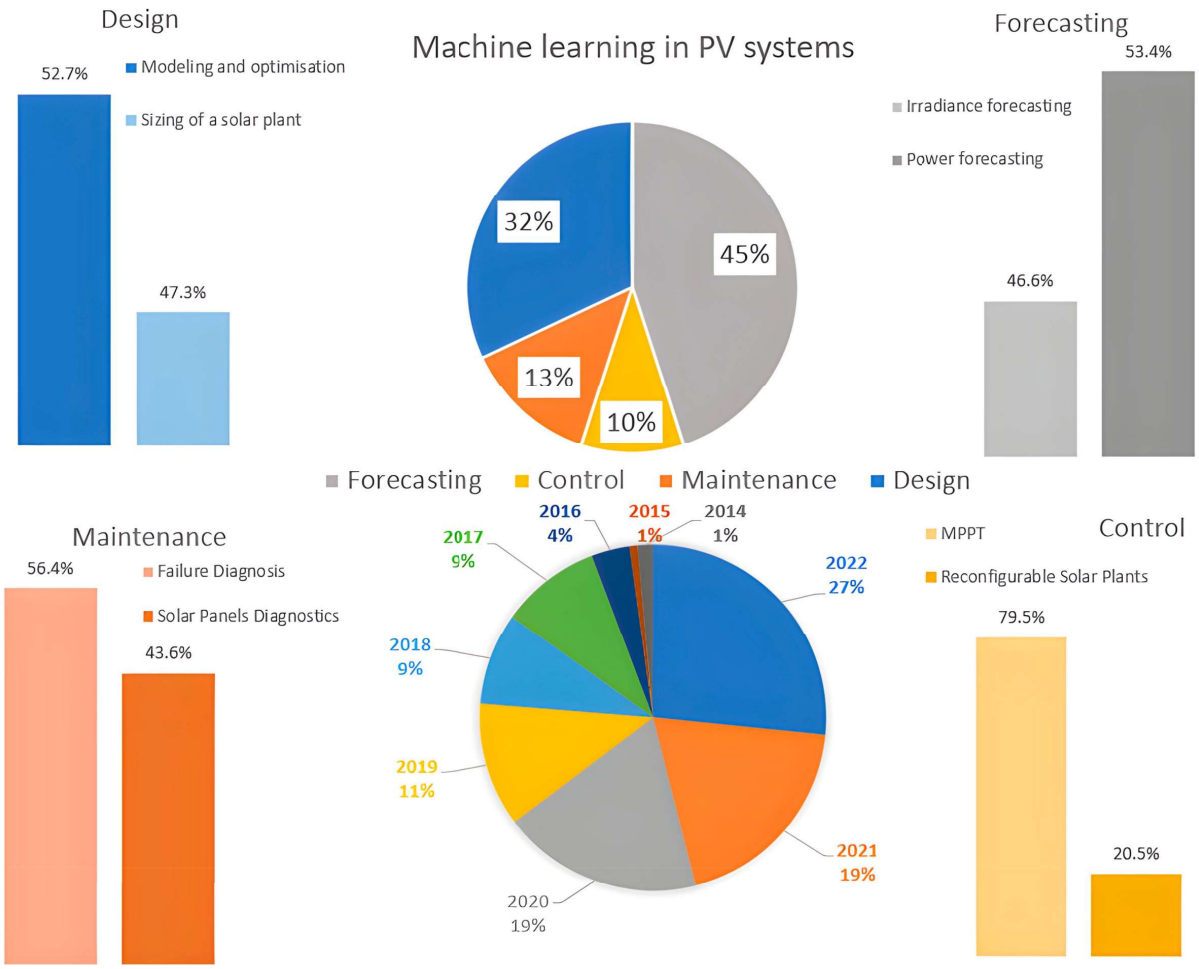


Figure 2.4: Distribution of ML technologies in the PV industry [11]

Globally installed solar power exceeded 1000 GW at the end of 2022 with up to 400 GW expected to be installed in 2023 alone [12]. With the rapid expansion of solar photovoltaic (PV) plants, robust and efficient inspection methods are crucial for detecting defects in PV modules and associated systems [13].

Thermographic imaging of PV modules, using thermal infrared (IR) cameras to detect temperature irregularities when compared to healthy ones, has emerged as a valuable inspection approach. Malfunctioning modules often exhibit temperature differences compared to healthy ones, making UAVs equipped with thermal cameras a very powerful tool for inspecting PV plants.

Traditional fault detection methods for PV modules often use classical computer vision techniques. These include binary thresholding for contour extraction and PV module segmentation, as well as obtaining texture features for classification with Support Vector Machines (SVMs). However, these methods have strong limitations. They heavily depend on manual assumptions and heuristics, requiring extensive hyperparameter tuning. They struggle to generalize to new images, making them less adaptable.

Lukas Bommes *et al.*[14] collected a dataset containing millions of RGB and thermographic video frames over seven large PV plants using a UAV equipped with GPS. They developed a tool, described in Figure 2.5, to map the PV panels using a Mask R-CNN to perform module segmentation and a ResNet-50 deep convolutional classifier to catalog the thermal anomalies in ten classes: Healthy, Open-circuit, Short-circuit, Substring open-circuit, Substring short-circuit, Module PID, Multi hot-cells, Single hot cell, Warm cell(s), Diode overheated and Hot spot. The study demonstrated significant improvements over state-of-the-art methods, achieving a test accuracy exceeding 90% across various plant domains without requiring hyperparameter adjustments.

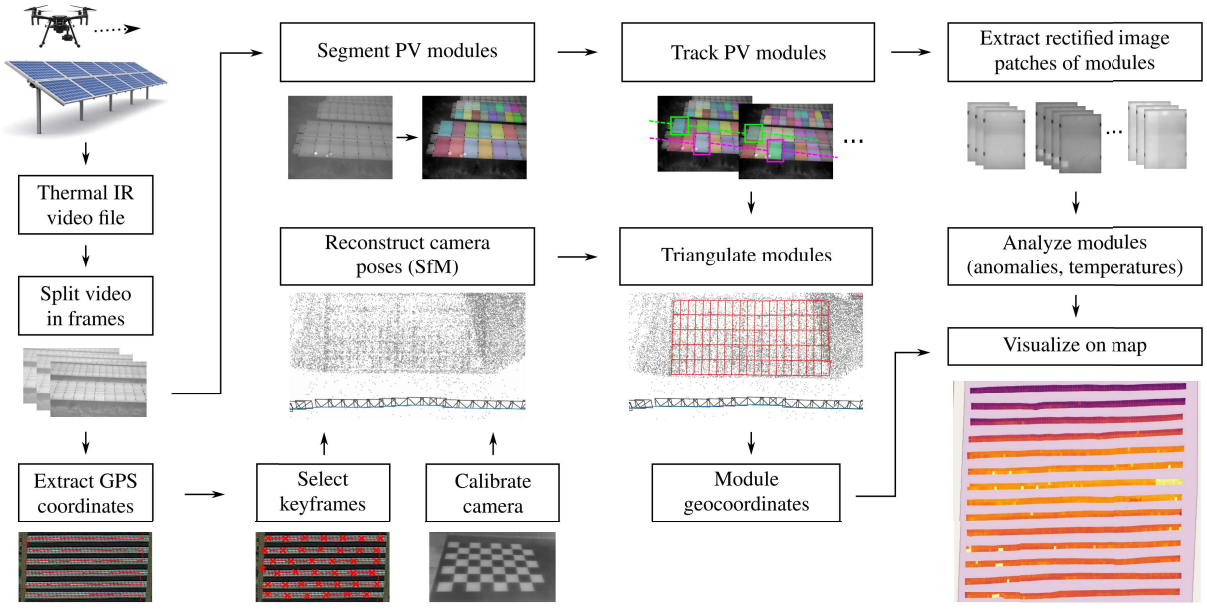


Figure 2.5: Process of PV modules inspection from aerial IR videos [14]

Noteworthy Inspect integrates computer vision cameras, edge computing devices and proprietary AI models onto existing fleet vehicles without any intrusive modifications, as shown in Figure 2.6. These cameras autonomously detect and geolocate utility poles during routine operations, capturing high-resolution images for each identified pole.

Figure 2.6: Noteworthy embedded inspect system [15]

These images serve as inputs for proprietary machine learning models, enabling diverse functionalities such as defect identification, asset inventories, attachment distance assessment, lighting evaluation, vegetation analysis, as-built assessments, and more. By combining stereo vision point clouds and embedded neural networks, their technology produces highly accurate 3D reconstruction images, providing precision comparable to LiDAR technology but without its typical time and cost requirements. Noteworthy Inspect offers a comprehensive utility inspection solution with its high-resolution image capture, rapid processing speed and cost-effective approach. It provides a cloud-based data visualization platform that seamlessly syncs inspection data from edge devices, granting access to pole coordinates, component details and identified defects. Additionally, it offers an API for smooth integration with existing geographic information systems (GIS), asset management, work order scheduling, and other record-keeping systems [15].

Van Nhan Nguyen, Robert Jenssen, and Davide Roverso [16] identified some of the main challenges in DL vision-based UAV inspection and proposed a set of solutions. Being the first challenge the lack of training data, they collected a total of 28,674 images with a resolution of 6048x4032. By cropping the bounding boxes, they managed to create three additional datasets, resulting in a total of 94,477 256x256 images. Recognizing the time-consuming nature of labeling a large dataset, they used data augmentation to address class imbalance. This included making several crops around the bounding boxes and applying operations such as flipping, mirroring, blurring, adding noise and zooming during training, increasing the dataset size by 12 times.

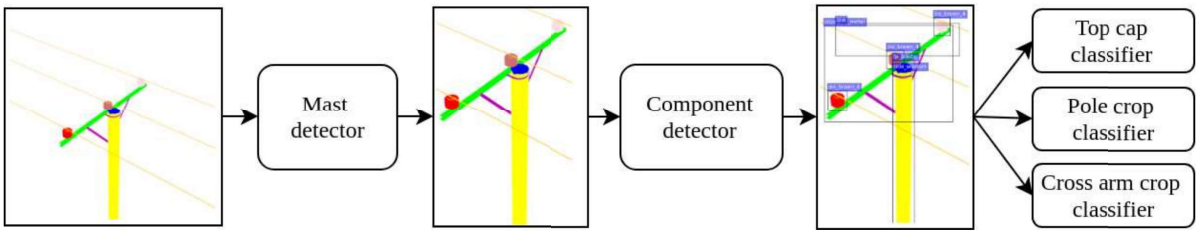


Figure 2.7: Multi-stage component detection and classification pipeline overview [16]

To address the challenge of detecting small power components and small faults, they proposed a multi-stage component detection and classification pipeline, with a general overview as illustrated in Figure 2.7. This pipeline consists of five components: a mast detector, a component detector, a top cap classifier, a pole crop classifier, and a cross arm crop classifier. It allows for a 'zoom-in' operation during inspection, enabling the detection of small faults on power line components(Figure 2.8), such as cracks on poles and cross arms, woodpecker damage on poles, and rot damage on cross arms.

Their best results in terms of mAP reached 81.3%, outperforming the other two methods in 7 out of 10 classes and achieving an inference speed of 300 images per minute.

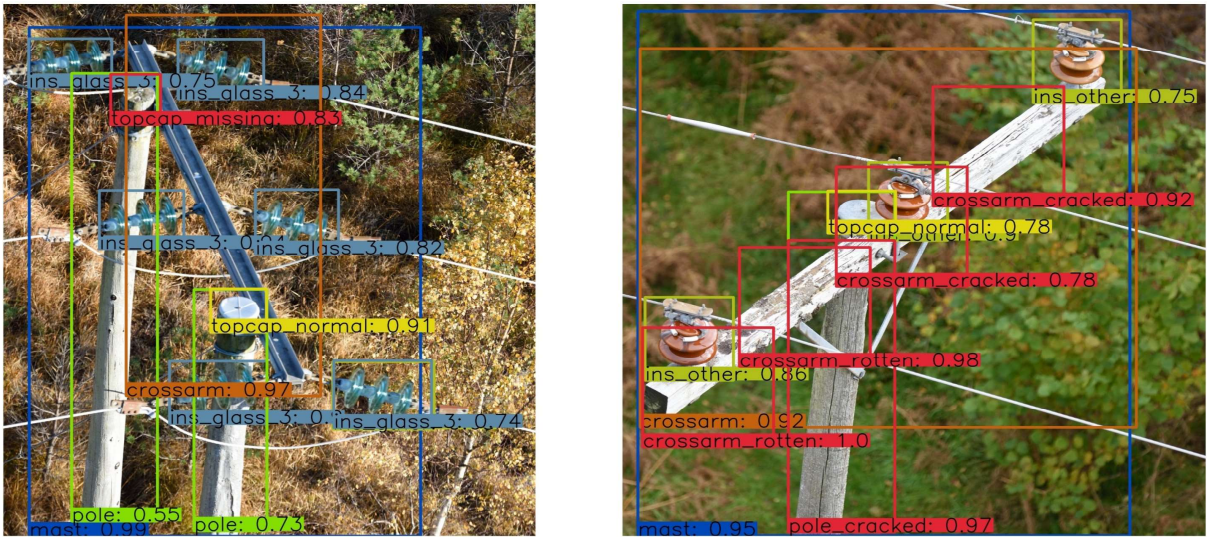


Figure 2.8: Multi-stage component detection and classification pipeline inference results [16]

They [16] also identified persisting challenges in automated visual inspections of distribution power lines. One such challenge is the identification of previously unseen components or faults, demanding comprehensive datasets. Detecting power line objects in intricate cluttered backgrounds poses another significant challenge, requiring sophisticated algorithms. Another critical aspect is the absence of robust evaluation metrics, necessary to gauge the accuracy and reliability of inspection systems effectively. To conclude, they discussed emerging solutions including integrating GPS mapping for self-driving UAVs to precisely map power line routes, using edge computing leveraged by embedded GPUs, for faster on-device data processing, enabling real-time analysis.

2.1.2 Electrical Grid Assets

In a 2023 global solar report, it was revealed that equipment-related underperformance led to losses amounting to $82 million for the analyzed 24.5 GW in 2022. Extrapolating this data to the entire solar industry indicates an annual loss of $2.5 billion [17].

Companies like Raptor Maps [17] are at the forefront of constructing an integrated operating system tailored for the solar industry, facilitating scalability and aligning with global climate objectives. The approach involves comprehensive solar PV inspections, reporting, and analytics, utilizing aerial thermography. Raptor Maps' inspection solution processes thermal images of PV systems, identifying, classifying, and prioritizing all anomalies. This precise anomaly location data, enables an efficient response, allowing PV systems worldwide to optimize productivity, by offering geo-referenced digital twins, accessible both in the office and the field, as presented in Figure 2.9.

This technology empowers various stakeholders, including asset managers, field teams, performance engineers, plant managers, and control rooms, by providing essential insights to operate solar assets at peak efficiency and ensure their long-term viability.

By integrating sensor data, inverter yield data, field notes, inspection findings, and more, users can obtain a comprehensive 360-degree view of their assets. The built-in analytics engine provides actionable insights, and the API facilitates seamless integration with Business Intelligence systems. The platform enables robust underperformance root cause analysis, fostering a clear understanding of asset performance and facilitating targeted optimization strategies. Power production analytics and equipment anomaly detection contribute to extracting actionable insights from solar site data.

The technology also enhances team efficiency through an intuitive mobile app. Field teams can easily locate and navigate to equipment requiring remediation, ensuring safe and efficient issue resolution. The app allows capturing notes, images, serial numbers, and more directly from the field. Additionally, it automates the generation of warranty claims packages using data available in the digital twin.

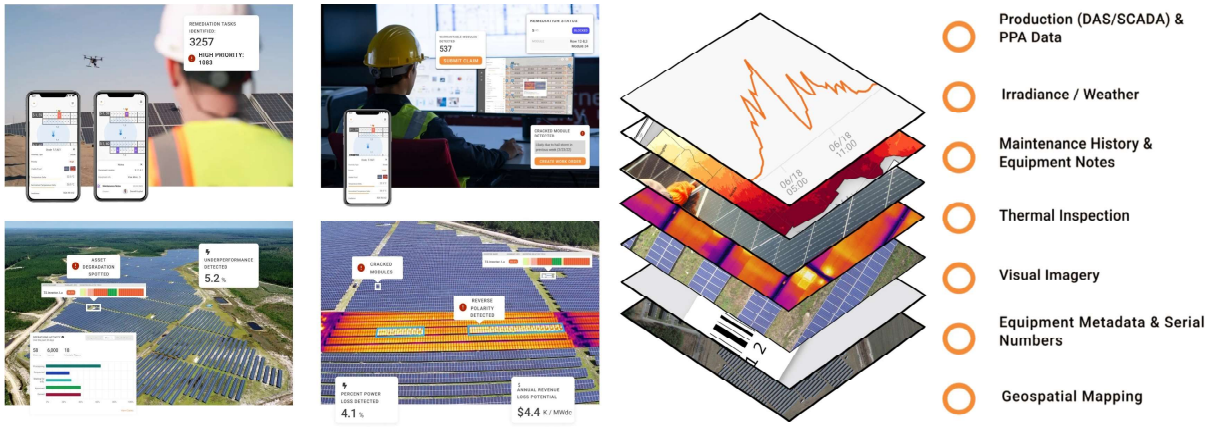


Figure 2.9: Process of PV modules inspection from aerial IR videos [17]

Companies like eSmart Systems Grid Vision focus on managing asset information and conducting virtual inspections for Transmission and Distribution (T&D). They performed a comparative analysis of two modes of visual inspection for overhead transmission lines: foot patrols and imagery-based virtual inspections [18].

Virtual inspections use high-resolution images from unmanned aircraft systems, while foot patrols involve systematic on-site inspections, documented on field computers. A 5-level priority rating system was adopted. P1 for immediate action, P2 requires near-term attention, priorities 3 and 4 are for scheduled maintenance, and priority 5 for defects to be monitored. Results presented in Table 2.1 indicate that virtual inspections detected over 60% more defects than foot patrols.

Table 2.1: Number of defects identified by foot patrol and by the virtual inspection [18]

Priority		New Defects	
Description	Level	Foot Patrol	Virtual Inspection
Action Required	P2	1	3
Scheduled Maintenance	P3/P4	117	170
Periodic Monitoring	P5	143	247
Total		261	420

The discrepancy in P5 defect counts between foot patrols (143) and virtual inspections (247) might arise from inspectors potentially focusing more on reporting actionable defects (P1-P4) while omitting P5 defects, which are solely monitored.

Certain defect types, as shown in Figure 2.10, are more easily detectable in virtual inspections due to their top-down views, capturing components that might be challenging to spot during foot patrols. Conversely, foot patrols with a botom-up view, excel in identifying others defects, as illustrated in Figure 2.11.

Figure 2.10: Examples of defects that are easier to detect during a virtual inspection [18]

Figure 2.11: Examples of defects that are easier to detect during a foot patrol [18]

The increasing volume of collected data such as images, during infrastructure inspections, and its processing, is currently being strangulated by human interpretation tasks, mainly because escalating this pipe-line element is expensive and prolonged. However, many inspection responsibilities can be automated using computer vision and deep learning techniques to support the system's growing demand.

The Electric Power Research Institute (EPRI) is leading a collaborative initiative to advance utility infrastructure inspection by sharing labeled datasets [19]. These datasets aim to support immediate and future research efforts.

Automated inspection workflows have the potential to revolutionize traditional methods, offering safety, quality, and efficiency benefits to both utilities and consumers. Improved awareness of diverse utility infrastructure can significantly reduce the likelihood of equipment failures during service. Visual inspection of high voltage electrical infrastructure is complex due to varied components and unique degradation modes in each area of interest. However, developing automated inspection technologies requires large datasets currently unavailable to handle real-world complexities.

EPRI aims to bridge this data gap by creating and sharing datasets, fostering both academic research and commercial innovation. For instance, by promoting development competitions on open-source platforms using labeled imagery, as shown in Figure 2.12.

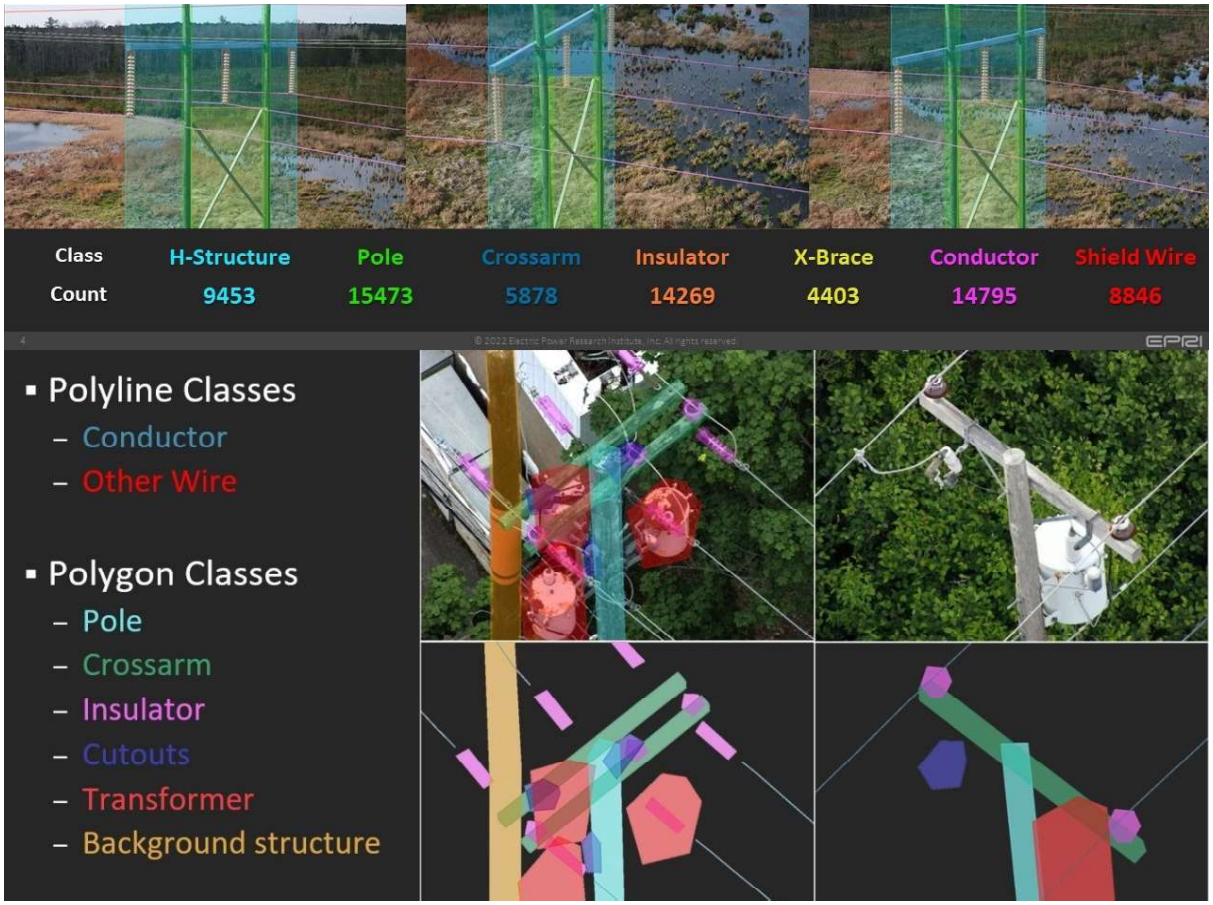


Figure 2.12: EPRI electric Transmission dataset and electric Distribution dataset [20]

2.1.2.1 Photo-Voltaic Panels

The main two common types of PV panels are crystalline silicon and Cadmium Telluride (CdTe) thin-film modules, with a configuration illustrated in Figure 2.13, and a characteristics' description summarized in Table 2.2:

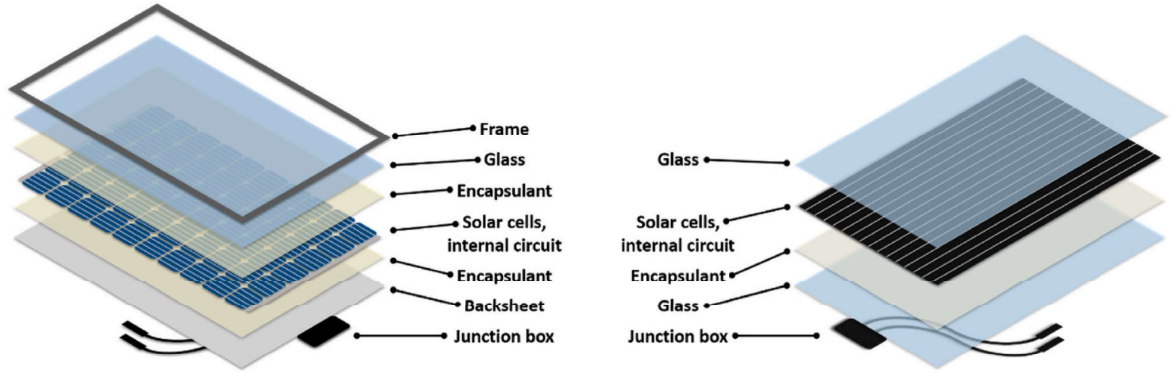


Figure 2.13: PV modules for crystalline silicon (left) and CdTe thin film (right) [21]

Crystalline Silicon Modules, made from crystalline silicon wafers, are renowned for their efficiency and durability. They exhibit higher conversion efficiencies, although they can be relatively more expensive to produce due to their manufacturing process. Their robustness and long-term reliability make them suitable for various environmental conditions, but their production can be energy-intensive, generating more waste. Crystalline silicon modules are rigid and heavier, typically used for stationary installations like rooftop solar panels.

CdTe Thin-Film Modules utilize a lighter and more flexible technology where a layer of cadmium telluride semiconductor material is deposited onto a substrate. Although historically less efficient than crystalline silicon modules, advancements have improved their efficiency. CdTe thin-film modules are often considered cost-effective due to their minimal material and energy requirements. They are more environmentally friendly and suitable for niche applications, like curved surfaces or portable solar panels. However, they may have a shorter lifespan and lower durability compared to crystalline silicon modules.

Table 2.2: Comparison of Crystalline Silicon and CdTe Thin-Film Solar Modules

Characteristic	Crystalline Silicon	CdTe Thin-Film
Material	Silicon wafers	CdTe thin film
Efficiency	Generally higher	Lower, but improving
Cost	Can be more expensive	Cost-effective
Durability	Robust and reliable	Generally less durable
Environmental Impact	Energy-intensive production	More environmentally friendly
Flexibility	Rigid	Lighter and more flexible

A PV system failure is defined by the photovoltaic components inability to function optimally and sustain their intended performance levels. These failures stem from a spectrum of external and internal causes. External factors include issues during transport, flawed installation practices, and exposure to extreme environmental conditions. These conditions encompass environmental stressors like excessive dust, soiling, or interference from vegetation, all of which can restrain the system's efficiency. Internal failures arise from structural issues within the components themselves. These internal causes involve the emergence of cracks within the solar cells or faults in cell connections. These defects often lead to diminished power generation, compromised efficiency, or, in severe cases, complete malfunction of the PV system [13].

The attraction of electrons from the N-type material towards the P-type material results in the formation of a depletion zone within the N-P junction, creating an equilibrium state of charged particles with a full valence band. Electrons within photovoltaic cells absorb energy from photons, elevating their energy level. This heightened energy enables the electrons, trapped in the depletion zone of the N-P junction, to transition from the valence band to the conduction band, thereby generating electron-hole pairs. As the absorbed energy increases the mobility of electrons within the depletion zone, they gain the ability to overcome the electric field force, migrating into the original N-type layer. This migration diminishes the size of the depletion zone and establishes an electric potential around the silver and aluminum poles of the photovoltaic cell [22].

Hot spots in PV panels occur when normal current flow within silicon semiconductor cells is disrupted, often due to blocked photoelectric effect areas, as shown in Figure 2.14. These blockages may stem from internal damage or pronounced irradiance contrasts caused by objects shading the cells. This reverses the current flow, causing shaded cells to act as resistors in the panel circuit, consuming energy instead of producing it, resulting in localized temperature increases.

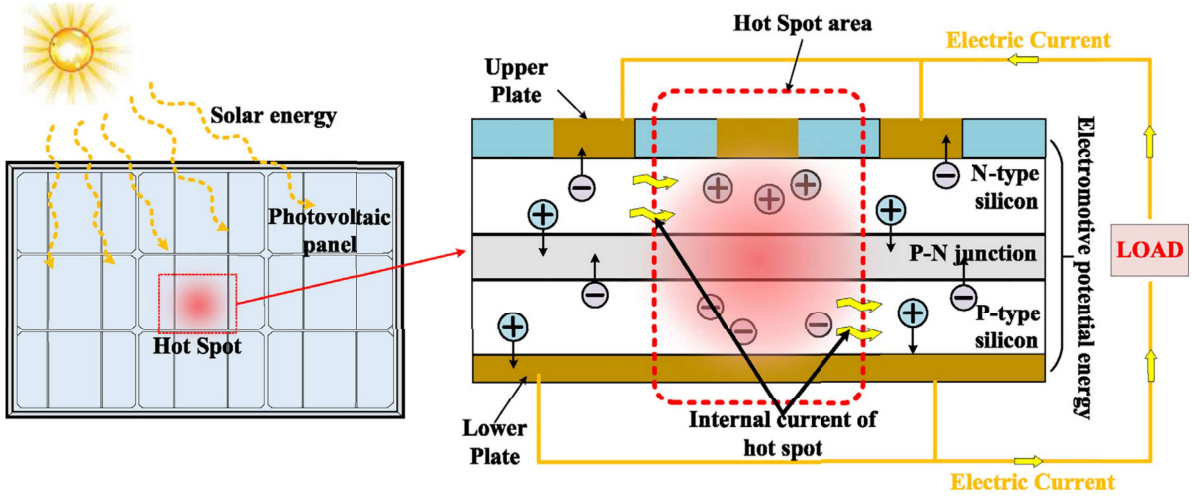


Figure 2.14: Formation process of hot spots of PV panels [23]

Prolonged exposure to hot spots can damage cells, potentially leading to burnout or degradation. Severe cases might cause cell failure, disrupting the entire panel's functionality. Hot spots not only reduce efficiency but also pose fire risks. Bypass diodes in solar panels mitigate partial shading impacts by redirecting current flow around shaded areas, preventing excessive heat buildup and reducing hot spot risks. Proper system design and monitoring are vital to promptly detect and address shading issues, ensuring safe and efficient PV array operation.

Figure 2.15 depicts the adverse effect on PV systems, specifically showing a scenario with vegetation-induced shading on a PV module. In this simulated scenario, the shaded module experienced a significant power drop of over 45%, declining from 365 W to 199 W. This reduction in output from a single module propagates throughout the entire string of connected modules, leading to substantial overall losses.

This illustration highlights the cumulative impact of shading across an entire installation consisting of series-connected modules. It's crucial to note that despite minor shading appearing insignificant, it does impact the system's overall performance by causing an efficiency reduction of up to 10%. Interestingly, in specific cases involving modules equipped with Half-Cut technology, the shading's impact tends to be relatively lower compared to standard modules [24].

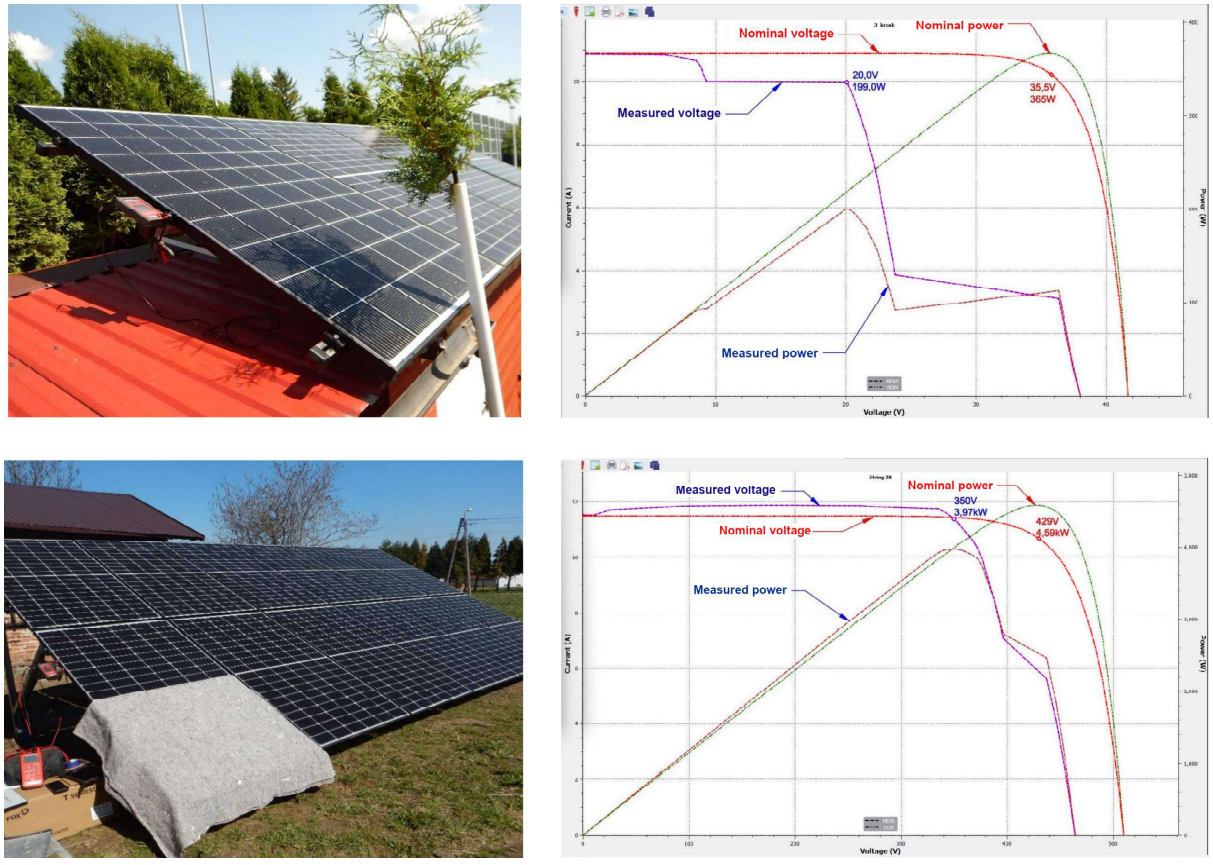
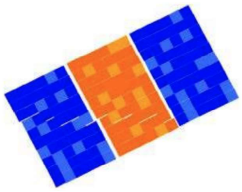
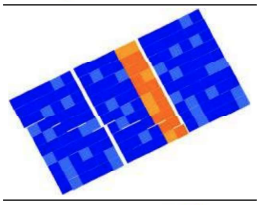
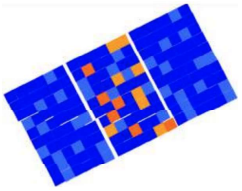
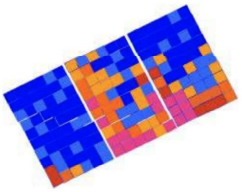
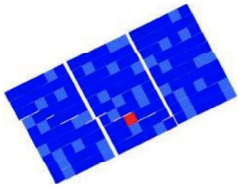
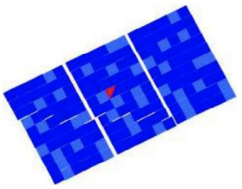
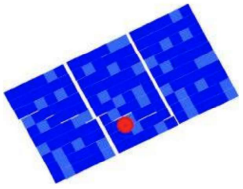
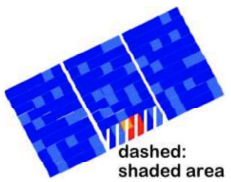


Figure 2.15: Shading PV module and PV panel and the respective I–V curves [24]

A summary of observed patterns in Photovoltaic module images from outdoor Infra-Red measurements, along with their descriptions, potential failure causes, and their impact on the electrical output, is presented in Table 2.3.

Table 2.3: IR image patterns observed of PV module possible failures [13]

Visual Pattern	Description	Potential Causes	Electrical Measurements	Remarks
	One module warmer than others	Module is OC, not connected to the system	Module normally fully functional	Check wiring
	One row (sub-string) is warmer than other rows in the module	SC sub-string OC sub-string Bypass diode SC, Internal SC	Sub-strings power lost, Reduction of V_OC	Burned spot at the module One Diode shunted
	Single cells are warmer, recognizable patchwork pattern	Whole module is short circuited All bypass diodes SC Wrong connection	Module power drastically reduced, Strong drop in V_OC	Check wiring All diodes shunted
	Single cells are warmer, lower parts and close to frame hotter than upper and middle parts	Massive shunts caused by Potential Induced Degradation and/or Polarization	Module power and FF reduced. Low light performance more affected than at STC	Change array grounding conditions Recovery by reverse voltage
	One cell clearly warmer than the others	Shadowing effects Defect cell Delaminated cell	Power decrease, not necessarily permanent, e.g. shadowing leaf	Visual inspection needed, Cleaning Shunted cell Delamination
	Part of a cell is warmer	Broken cell Disconnected string interconnect	Drastic power reduction FF reduction	Cell cracks Burn marks Interconnects
	Pointed heating	Artifact Partly shadowed Bird dropping	Power reduction, dependent on form and size of the cracked part	Crack detection after detailed visual inspection Possible cell cracks
	Sub-string part remarkably hotter than others when equally shaded	Sub-string with missing, or OC bypass diode	Massive I_SC and power reduction when part of this sub-string is shaded	May cause severe fire hazard when hot spot is in this sub-string

2.1.2.2 Power Line Insulators

The problem of reliable suspending high voltage transmission line conductors has to deal with mechanical, electrical and environmental extreme stresses.

The air present in the atmosphere and surrounding the bare naked steel-core aluminum conductors is an optimal dielectric choice, admitting that the electric stress is kept below the ionization threshold. However, to suspend overhead power lines, solid materials are used as anchor points.

The classification of the existent types of insulators is presented in Figure 2.16 [25].

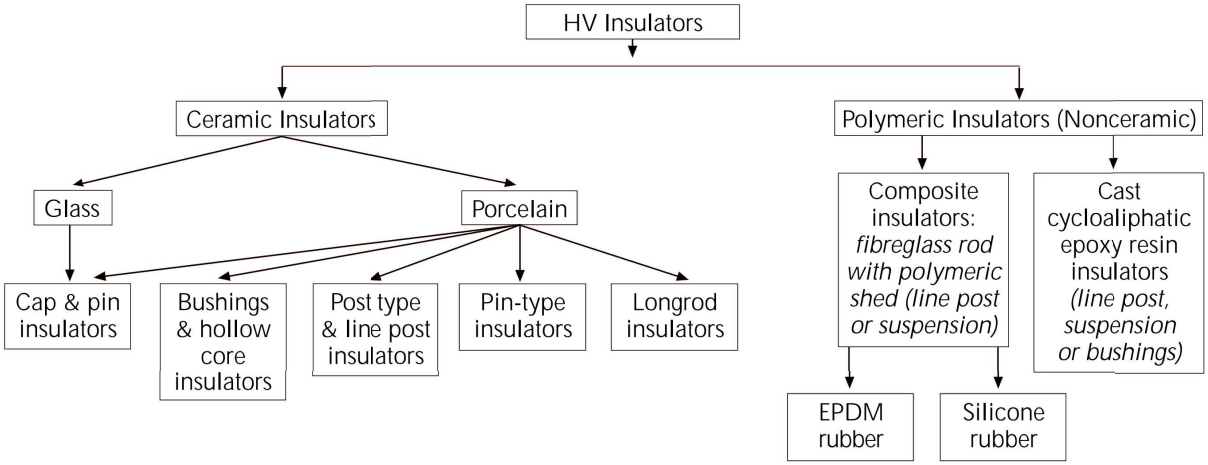


Figure 2.16: Classification of power lines insulators [25]

The main used insulator type is the *Cap and Pin* disc, made of ceramic [26] or glass materials, represented in Figure 2.17. Each unit is connected to another by sliding the steel pin in the steel cap cavity [27], forming an insulator string that typically support loads of 120 kN (12 ton) of mechanical tension.

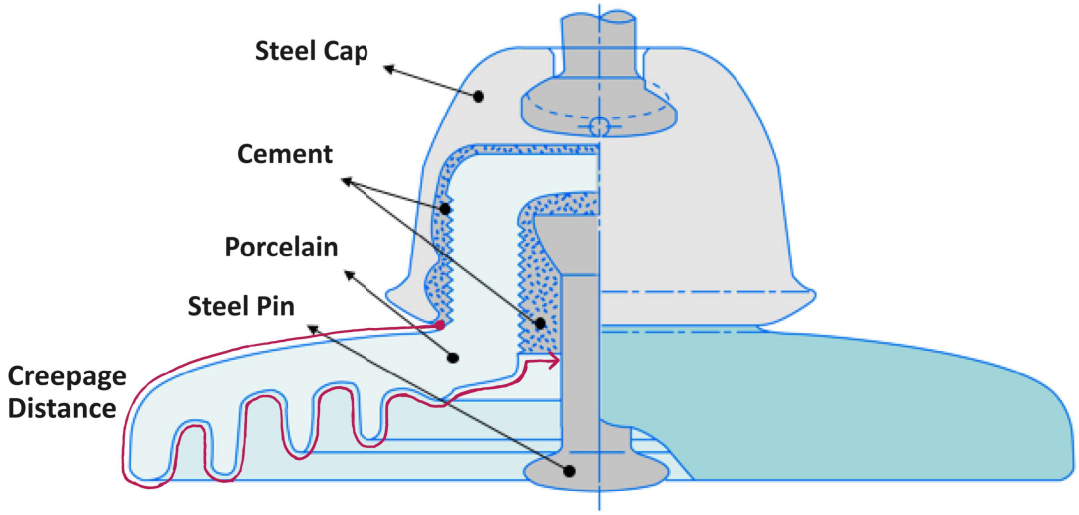


Figure 2.17: Typical Cap and Pin disc insulator [28]

When exposed to the environment, the hydrophobic surface of ceramic insulators becomes contaminated by coastal pollution, including sand and salt carried by sea winds, or other deposits from industrial sources. These contaminants increase surface conductance by forming a conducting electrolyte, especially in moist weather conditions. While light rain and fog exacerbate the issue, heavy rain helps in removing some debris. The choice of the insulators specific creepage length depends on the severity of pollution at the site [29].

Ionization can occur in the air if the electrical field strength exceeds 3 kV/mm at atmospheric pressure. The voltage required to cause flashover in a clean, dry single cap&pin insulator with a 280 mm creepage distance is 72 kV. However, under medium pollution conditions, it is estimated that its performance decreases tenfold [30]. This significant reduction in performance is attributed to the presence of a conducting layer on the insulator's surface.

Leakage current flowing over the insulator surface, along with its heating effect, causes the formation of dry bands or stains in specific spots, as Figure 2.18 shows.

Figure 2.18: Flashover damages exemplification [31]

The manufacturing process of glass insulators includes a thermal cooling treatment that maintains their mechanical strength while deliberately creating a clearly visible shattered disc in faulty insulators after they've been electrically punctured.

Table 2.4: Insulators Operational Defects and Symptoms

Operational Defect	Potential Causes	Visual Symptoms
Contamination	Sand and Dust Salt and Pollution Bird Droppings	Stains on Surface UV Glow under certain conditions
Flashover	Moisture and Contamination Lightning Strikes	White Bands or Marks Damaged Coating
Broken or Cracked	Mechanical Stress Aging of Material Rapid Thermal Changes	Sharpened Shape Core Exposure Disc Absence
Metallic Corrosion	Aging of Metal Parts Pollution and Contaminants Harsh Weather Conditions	Rust Coloration Texture Changes due to Rust Drained or Corroded Appearance

2.1.3 Quality Control

Quality Control (QC) implies more than just detecting defects, it also involves finding the root causes of those defects. To ensure that the output meets the required standards, in addition to perform inspections, tests and analysis to identify and categorize defects in products or processes is essential to understand and determine the underlying defect causes by investigating the entire production process, allowing the implementation of corrective actions to address the identified issues.

Quality control is not a one-time activity, it is an ongoing process of continuous improvement, as illustrated in Figure 2.19, where lessons learned from defect detection and root cause analysis are used to refine processes

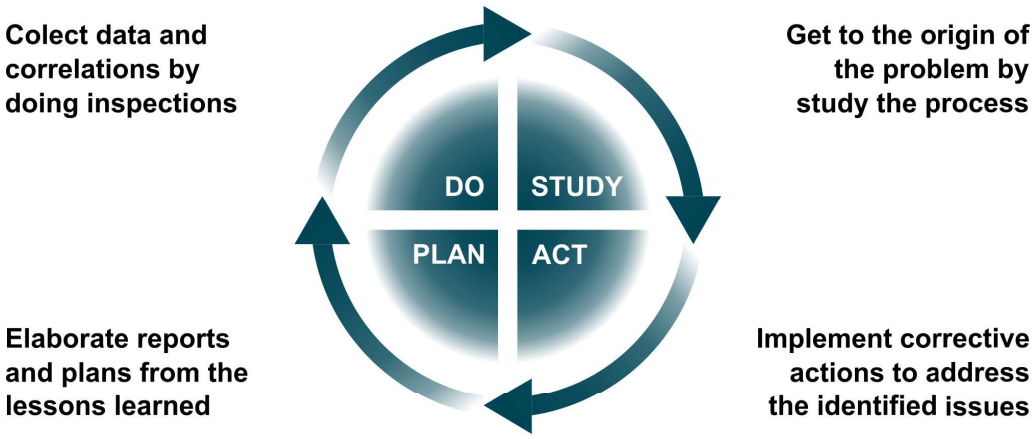


Figure 2.19: Continuous improvement cycle of Quality Control

The inspection process plays a fundamental role in the continuous improvement cycle by providing valuable data that feeds into the other stages. Tasks like dimensional inspection, imply controlling the shape and the physical measures of a product according to its specifications, which include its respective tolerances. Tasks like visual inspection imply comparing color, texture, brightness, opacity and other properties, to the required standard, within a proper quality range. Human labor performing those tedious repeated tasks and procedures introduces variances in quality control. In this industrial growing world, time is an increasing valuable resource and technology-based solutions have assisted humans in creating robust and efficient methods to benefit both employers and employees in continuous manufacturing operations.

The need for computer vision in quality control has been driven by the increasing complexity and precision requirements of modern manufacturing. Products have become more intricate, and consumers demand higher quality standards. Moreover, regulatory requirements have become stricter in various industries, making compliance a top priority. Computer vision technology has provided a solution to meet these challenges by automating inspections, improving accuracy, and enhancing overall quality control processes, thus ensuring the production of high-quality and compliant products [32].

2.1.3.1 Visual Inspections

Often considered the most basic and direct form of Non-Destructive Testing (NDT) methods, Visual Inspection is used to evaluate the condition of an asset as part of the overall maintenance process [33].

This inspection method comprehends the analysis of naked eye visual spectrum information, dismissing special equipment but requiring special training. The observer visual perception is used to identify visible defects, anomalies, wear and tear, or other forms of damage, such as cracks, corrosion, leaks, or irregularities in a product or structure that could potentially impact its functionality, safety, or quality.

The primary goal of an inspection is to prematurely identify minor issues and their causes in order to prevent more significant and costly damage or consequences. Visual inspections are commonly used across various industries, including manufacturing, construction, maintenance, and quality control.

While visual inspections are valuable for detecting surface-level issues, they may not uncover defects hidden beneath the surface. As a result, they are often complemented by other NDT techniques, including Radiography or Thermography, which expand the observability of the electromagnetic spectrum; Magnetic Testing or Eddy Current Testing, which is employed to identify structural imperfections such as cracks and voids by examining disruptions in the flow of the magnetic or electric field; Acoustic or Ultrasonic Testing, which involves comparing the material's sound propagation with a reference or baseline signal obtained from a defect-free asset; and others more [34].

Advancements in technology have eliminated the need for in-person visual inspections. Digital cameras make it possible to collect large amounts of high-detail data locally, which can then be efficiently analyzed remotely. This is particularly useful in harsh environments such as exploration mines, storage tanks, or industrial boilers.

To broaden the scope of visual inspections, drones and other robots are utilized to access and collect visual data from remote locations such as wind turbine plants, offshore platforms, or submersible structures, as well as confined spaces like pipelines, ducts, or pits. Figure 2.20 exemplifies the environments where these tools prove beneficial.

Figure 2.20: Aerial, confined, and submarine inspections by unmanned vehicles [35] [36]

2.2 Computer Vision

Computer vision is a scientific field that studies means of enabling artificial vision. Its origin can be traced back to the invention of the photographic camera in the 19th century, followed by the television. However, the term 'computer vision' only began to be used as a concept to imitate the human visual system in the 1960s. The surge in studies on image processing algorithms emerged in the 1970s, with the concept of optical character recognition (OCR) being developed in the 1980s [37].

Computer vision (CV) has been evolving since then, at an exponential increasing rate. Nowadays, it's challenging to name an application field that is not served by this technology, ranging from the military to medical sectors, as well as industry and mobility. CV systems are technological tools often used to expand human vision's limitations and exceed their capabilities. One primary goal is the translation of information between domains by processing signals from outside the visual electromagnetic spectrum into a format for human visual interpretation. Examples of such processes include technologies like SoNAR, RaDAR, LiDAR, MRI scans, meteorological satellites, and radio telescopes. Another objective of computer vision systems is to overcome the biological and physiological limitations of humans, such as fatigue and subjectivity, by emulating the entire visual processing and automating the generation of solutions in the problem's counter-domain.

One of the mainstream CV projects today is in autonomous vehicles, where CV technology plays a pivotal role by feeding motion control systems with data from the environment interpretation trough different techniques such object detection, instance segmentation, semantic segmentation, or point cloud detection (Figure 2.21).

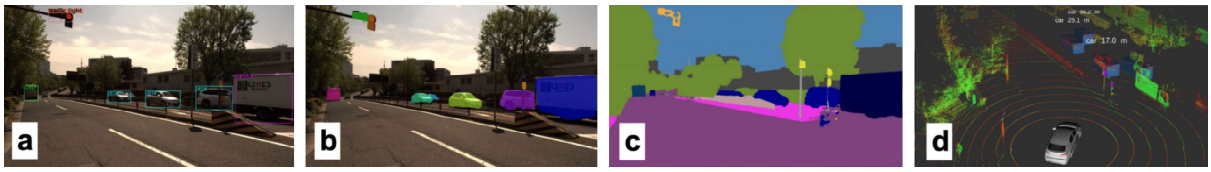


Figure 2.21: Different techniques for environment interpretation [38]

Other highly innovative CV projects in the Augmented Reality (AR) scope deal with utilizing specific wavelengths and enhancing object edges to enable scenarios like aiding firefighters in navigating trough smoky environments (Figure 2.22) [39].

Figure 2.22: Firefighting augmented reality scenario [39]

An increasingly captivating topic that has garnered significant scientific interest involves the direct integration of a computer vision system into the human visual system by developing spherical artificial eyes as bionic prostheses, as illustrated in Figure 2.23. Despite the incorporation of color vision, embedded preprocessing, and optical adaptability into spherical artificial eyes has consistently presented a formidable challenge, decoding the neural signals for visual perception might be the most complex task [40].

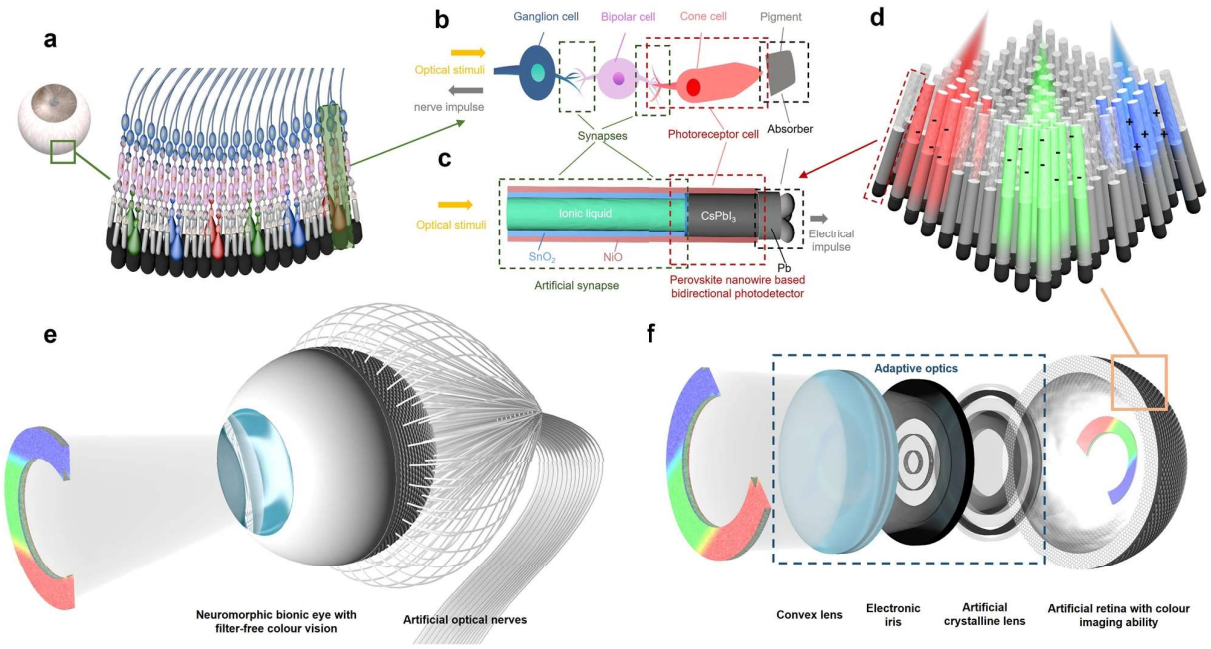


Figure 2.23: Neuromorphic bionic eye using a hemispherical nanowire array retina [40]

Computer vision has experienced significant advancements, primarily due to recent breakthroughs in Artificial Intelligence (AI), driven by an increase in both of the terms, the computational power and the available data.

The Homo-sapiens brain has undergone more than 300.000 years of genetic evolution, attaining a complex and intricately interconnected system responsible for a wide array of cognitive visual functions. This intricacy poses challenges in framing comprehensive formal specifications that capture visual processing tasks apparently easy for humans.

Deep learning techniques have transformed computer vision by enabling machines to learn and recognize intricate patterns and features without explicit knowledge of the underlying functions. Convolutional Neural Networks (CNNs) have accomplished remarkable results due to their computational efficiency and patterns encoding.

The future trajectory of computer vision holds the promise of further advancements, largely driven by the symbiotic relationship between AI and CV. While machines can excel in specific visual tasks, the adaptable nature of human vision remains a complex challenge. Researchers are continually striving to enhance the capabilities of computer vision systems, making them more versatile and context-aware of real-world scenarios.

2.2.1 Human Visual System

The concept of vision is more than just sight, the acquisition of images. Vision is the process of deriving meaning from what is seen [41].

Vision is the human's dominant sense, researchers estimate that more than eighty percent of our perception and cognition are mediated through vision, comprising a complex set of functions that involve a multitude of learnable and developable skills such:

- **Visual Skills:**
  - **Visual Acuity:** Measurement of image clearness, focus, and sharpness;
  - **Visual Field:** Encompasses the range of central and peripheral vision.
- **Visual Motor Abilities:**
  - **Alignment:** Refers to the positioning of the eyes and any deviations;
  - **Fixation:** Involves the ability to gaze steadily and accurately at an object;
  - **Pursuits:** Entail the ability to smoothly and accurately track a moving object;
  - **Saccades:** Imply the ability to quickly scan and change between objects;
  - **Accommodation:** Focusing on objects at various distances and changing focus accordingly;
  - **Convergence:** Aiming at an object and tracking it as it moves closer or farther away;
  - **Binocularity:** Integrates both accommodation and convergence;
  - **Stereopsis:** Assume the depth perception.
- **Visual Perception:**
  - **Visual-Motor Integration:** Assumes the coordination of eye-hand, eye-foot, and eye-body movements;
  - **Visual-Auditory Integration:** Relating and associating visual and auditory information;
  - **Visual Memory:** Remember and recall information based on what was seen;
  - **Visual Closure:** Mentally complete a visual picture when only some parts are visible;
  - **Spatial Relationships:** Understanding one's position in relation to objects and space, as well as the relative positions of objects to one another;
  - **Figure-Ground Discrimination:** Differentiation between the main object and its background.

In the process of vision, light from an object enters the eye and passes through the cornea, which bends and focuses the light onto the lens. The pupil controls the amount of light entering the eye. The lens then further focuses the light onto the retina, creating an inverted image. To accommodate both distant and near objects, the lens changes its shape, a process known as accommodation. Photoreceptors located on the retina generate electrical signals that are transmitted to the brain through the optic nerve. The brain interprets these signals and re-inverts the image, allowing us to perceive an upright image, as illustrated on the Figure 2.24.

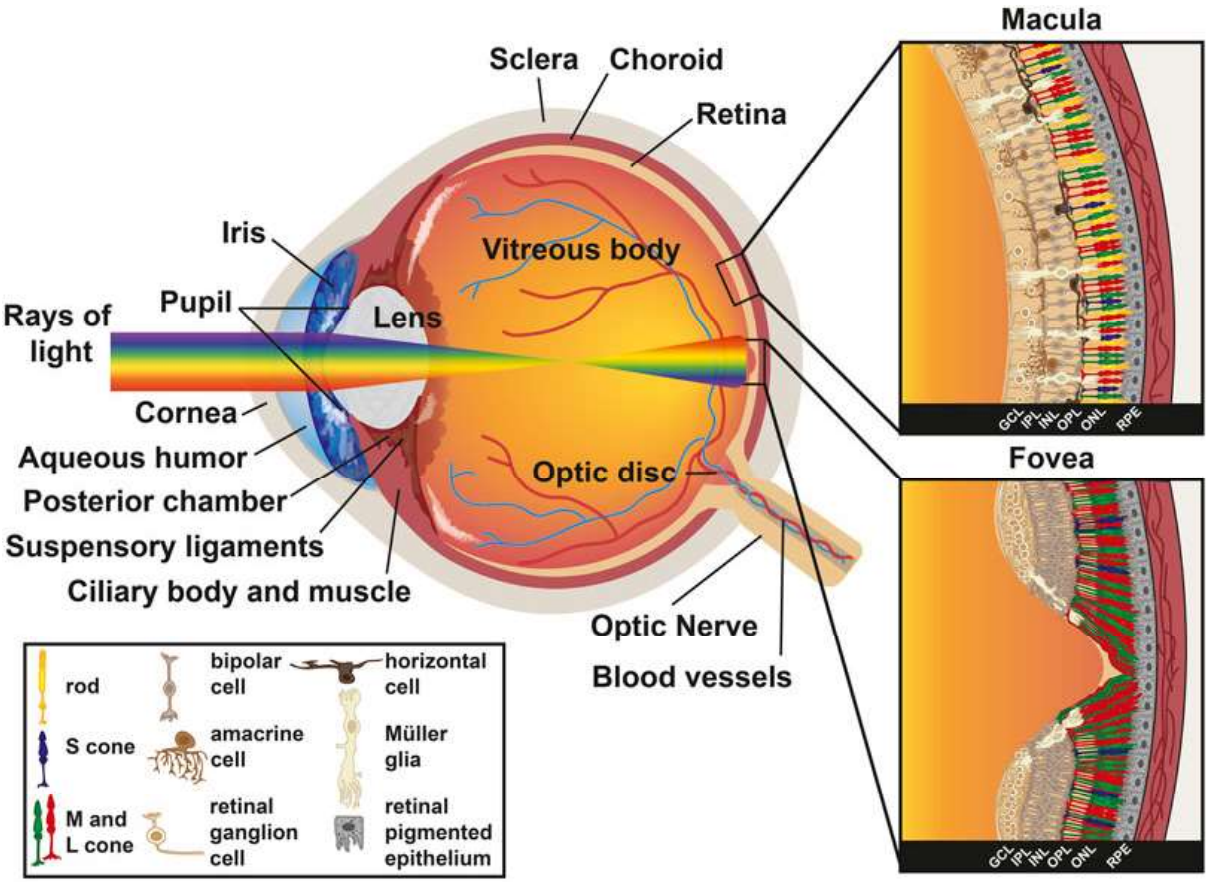


Figure 2.24: Photoreceptors in the Human Retina [42]

Within the retina, there are two distinct types of photoreceptor cells: rods and cones. Rods are highly light-sensitive and excel in low-light conditions, such as nighttime, but do not detect color. They are distributed throughout the retina, with a higher concentration in the periphery, and provide lower visual acuity. In contrast, cones require more light to function efficiently and are responsible for color vision in bright lighting. Three types of cones (red, green, and blue) enable us to perceive a wide range of colors. The fovea provides us with high-resolution, detailed vision in the center, where color discrimination is essential, while our peripheral vision enhances our ability to detect motion and low-light objects in the surrounding environment.

The initial stages of visual processing occur within the retina, where processes like color and edge detection are carried out by ganglion cells. The human brain is inherently wired for vision, a fact exemplified by intriguing experiments demonstrating that individuals who are medically blind can navigate around objects without consciously perceiving them, a phenomenon known as blind sight. The underlying explanation lies in the brain's ability to continue receiving input from the retina even when the primary visual processing systems are impaired. Thus, while the conscious visual processing part of the brain may not register the information, other regions of the brain could still provide spatial awareness [37].

Human vision perception is a remarkably intricate process, far beyond a simple translation of retinal stimulation. The brain undertakes extensive processing of the raw visual data received from the eyes to construct a unified, three-dimensional visual scene. This intricate process involves distinguishing foreground from background, recognizing objects from various perspectives, and interpreting spatial and temporal information. It operates under diverse lighting conditions while accurately processing spatial and temporal aspects of objects in the visual field.

At the core of this process lies the visual cortex of the brain, a critical area within the cerebral cortex specifically dedicated to processing visual information. Sensory input originating from the eyes converges on the visual cortex, with the primary visual cortex, also known as visual area 1 (V1), receiving this sensory input. Notably, both hemispheres of the brain house a visual cortex, with the left hemisphere processing signals from the right visual field and the right hemisphere handling signals from the left visual field, as shown in Figure 2.25.

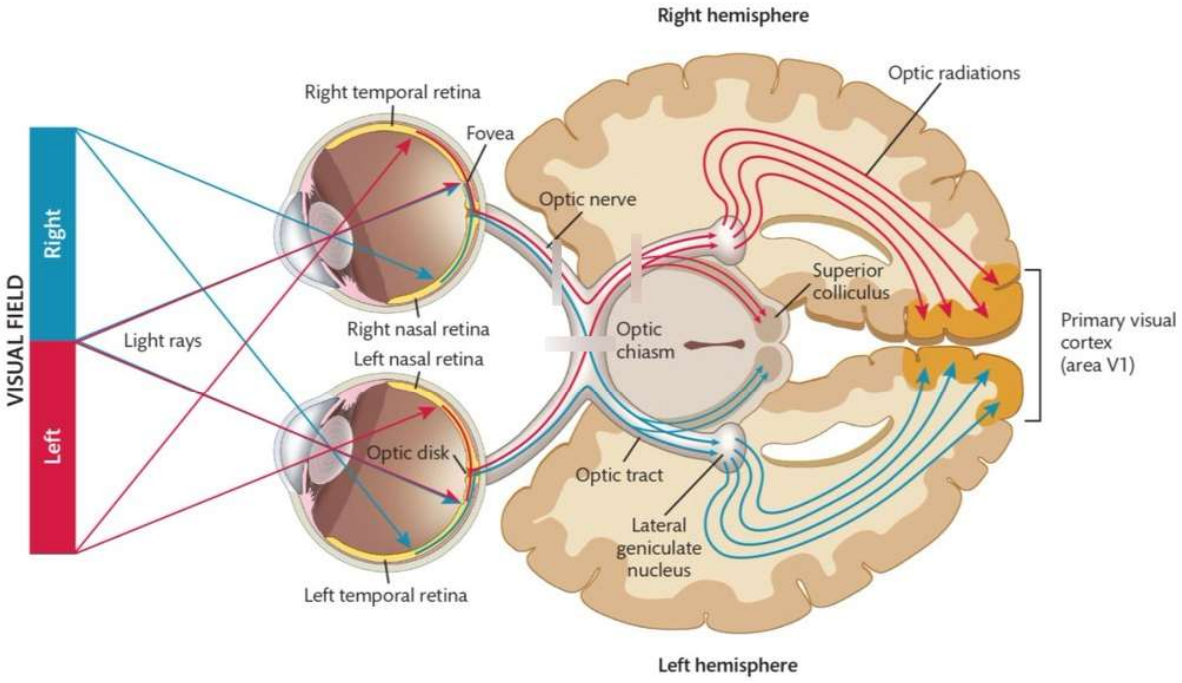


Figure 2.25: Main Pathways from Retina to Brain [43]

Receiving input from the ipsilateral lateral geniculate nucleus, the visual cortex enclose distinct specialized regions located around the calcarine fissure in the occipital lobe, each with unique functions and spatial characteristics, as depicted in the Figure 2.26.

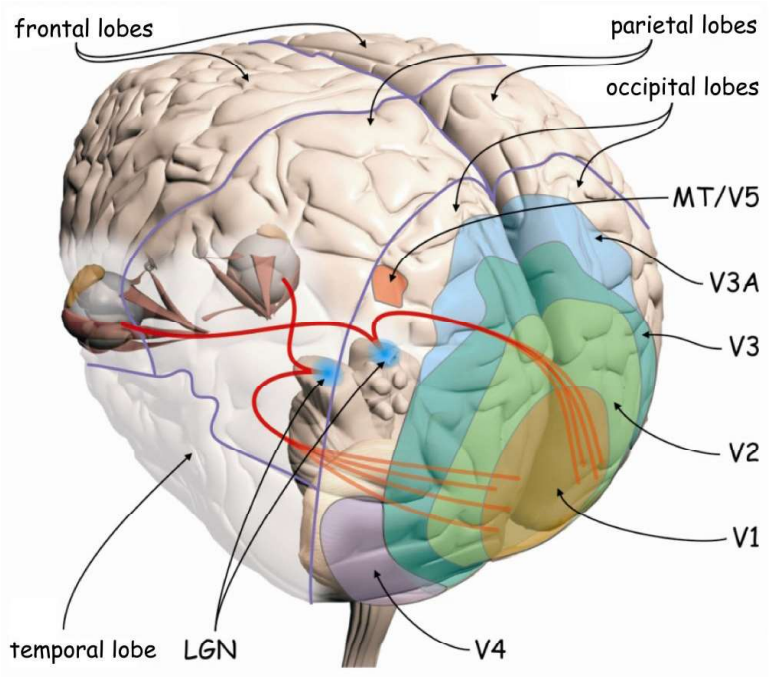


Figure 2.26: Visual Cortex spacial representation [44]

- **V1 (Primary Visual Cortex):** The starting point for processing visual information, receiving input directly from our eyes, assuming basic visual processing like detecting edges, shapes, and basic motion;
- **V2 (Secondary Visual Cortex):** More complex pattern recognition and identification of advanced visual features;
- **V3 (Visual Area 3):** Information related to object features, although its specific functions can vary among individuals;
- **V4 (Visual Area 4):** Conceiving understanding of visual details and intermediate complexity object features, such as geometric shapes;
- **V5 (Middle Temporal Visual Area or MT):** Perceiving of motion, iterating from local motion signals into our overall perception of movement. It also plays a key role in guiding eye movements based on visual motion cues;
- **V6 (Dorsomedial Area):** Provides a topographically organized representation of our entire field of vision. V6 responds to visual stimuli associated with self-motion and processes wide-field visual stimulation.

These distinct regions within the visual cortex collaboratively decode the visual world, encompassing basic shape detection, complex motion perception, and object feature recognition, facilitating the comprehensive understanding of the visual environment.

2.2.2 Digital Image Processing

One of the earliest applications of digital images was in the newspaper industry, when pictures were first sent between London and New York. The introduction of a submarine cable picture transmission system in the early 1920s, reduced the time required to transport a picture across the Atlantic from more than a week to less than three hours.

Digital Image Processing (DIP) methods can be categorized into two broad groups: methods where both inputs and outputs are images, and methods where inputs may be images, but outputs consist of attributes extracted from those images. The methods detailed in this section are interrelated, as suggested by the diagram in Figure 2.27.

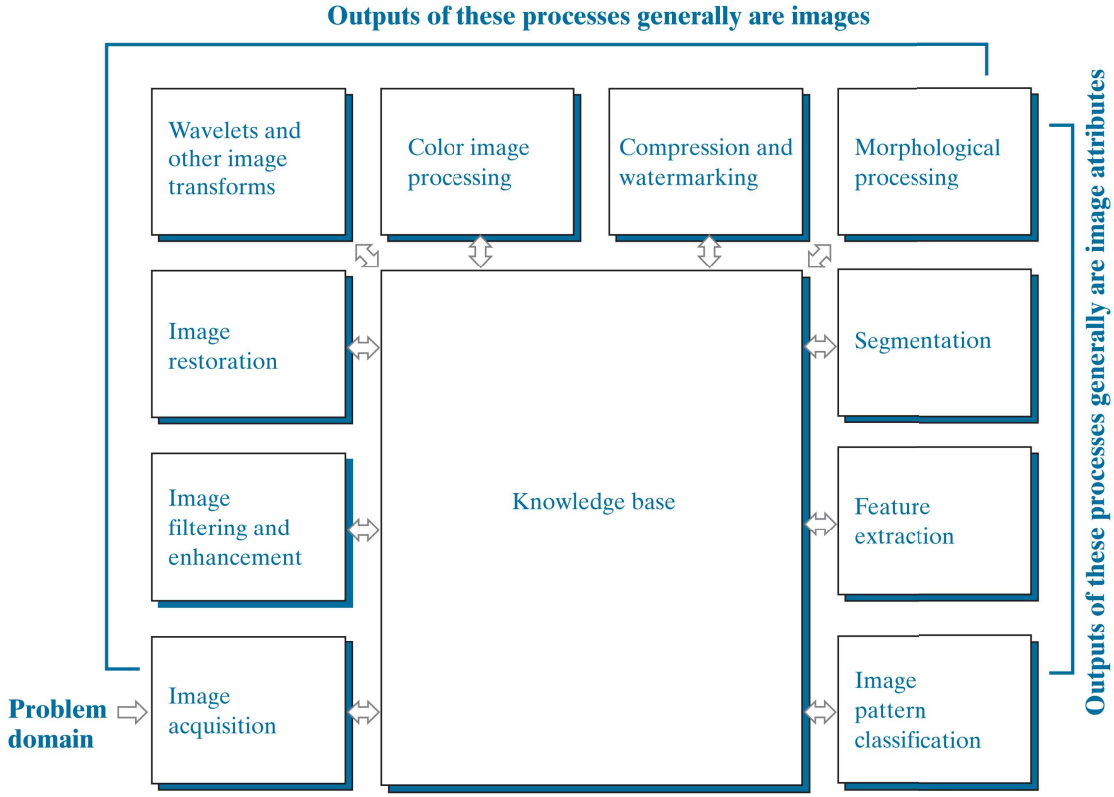


Figure 2.27: Fundamental steps in image processing [32]

Knowledge base comprehends information about a specific problem domain, allowing the system to make informed decisions during various stages of image processing. The complexity of the knowledge base can vary, ranging from simple details about the location of information in an image to intricate lists of potential defects or comprehensive image databases for specialized applications like change detection in satellite imagery. As the complexity of the image processing task increases, the demand for sophisticated processes and algorithms also rises. This highlights the importance of continuously advancing both the knowledge base and the processing techniques to address the challenges posed by intricate image analysis problems. The interplay between prior knowledge and computational methods is a dynamic aspect of image processing [32].

Image acquisition is the initial process, which can be as straightforward as receiving an image already in digital form. Generally, the image acquisition stage involves pre-processing, such as scaling, either in spatial resolution or intensity levels.

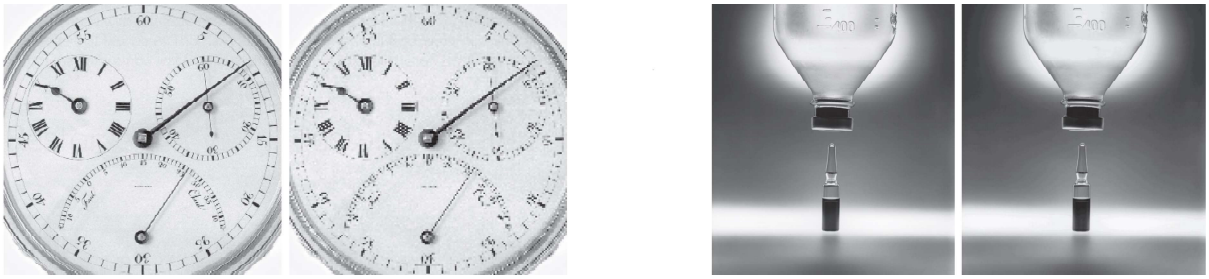


Figure 2.28: Effects of reducing spatial resolution (left) and intensity levels (right) [32]

Image enhancement is the process of manipulating an image so that the result is more suitable than the original for a specific application. Image filtering and enhancement and techniques are problem-oriented. For instance, a method useful for enhancing X-ray images might not be the best approach for enhancing satellite images taken in the infrared band of the electromagnetic spectrum. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works.

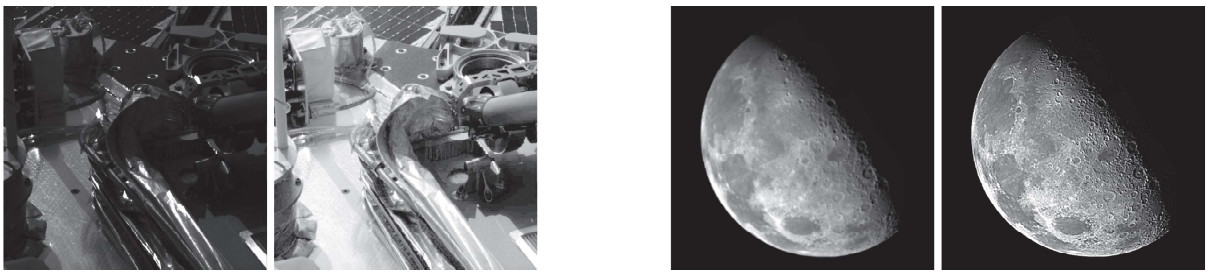


Figure 2.29: Effects of histogram equalization (left) and Laplacian filtering (right) [32]

Image restoration also aims to improve the appearance of an image, but unlike enhancement, which is subjective, image restoration is objective. Restoration techniques are typically based on mathematical or probabilistic models of image degradation. In contrast, enhancement is based on human subjective preferences regarding what constitutes a "good" enhancement result.

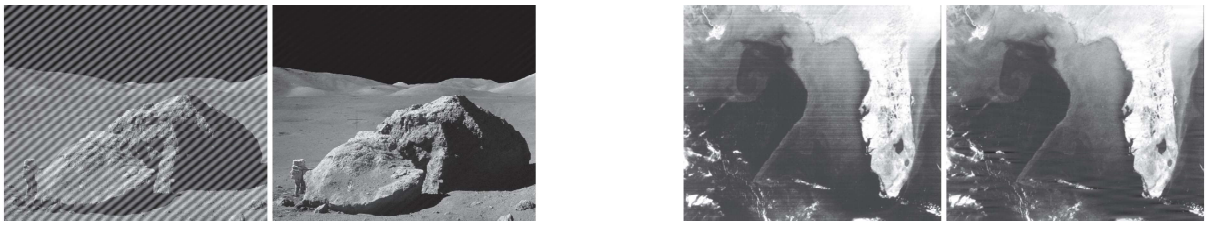


Figure 2.30: Result of notch reject a FFT point (left) and a vertical band (right) [32]

Image transformations form the foundation for representing images at various resolutions, particularly for image data compression and pyramidal representation, where images are subdivided successively into smaller regions. Additionally, there are other transforms used routinely in image processing, mostly based on the Fourier transform.

Color image processing has gained importance due to the significant increase in the use of digital images over the Internet. It involves fundamental concepts in color models and basic color processing in a digital domain. Color is also used as the basis for extracting features of interest in an image.

Compression, as the name implies, deals with techniques for reducing the storage required to save an image or the bandwidth needed to transmit it. Although storage technology has improved significantly over the past decade, the same cannot be said for transmission capacity, especially in Internet uses characterized by significant pictorial content. Image compression is familiar, perhaps inadvertently, to most computer users in the form of image file extensions, such as the jpg file extension used in the JPEG (Joint Photographic Experts Group) image compression standard.

Morphological processing involves tools for extracting image components useful in the representation and description of shape. This marks the transition from processes that output images to processes that output image attributes.

The word 'morphology' commonly denotes a branch of biology that deals with the form and structure of animals and plants. In the context of mathematical morphology, the same word is a tool for extracting image components useful in the representation and description of region shape, including boundaries, skeletons, and the convex hull. Morphological techniques mark the transition from methods with inputs and outputs as images to methods with outputs as image attributes, serving tasks such as object extraction and description. Morphology aligns with other tools like segmentation, feature extraction, and object recognition, forming the foundation of techniques for extracting 'meaning' from an image.

Segmentation of images is defining segmented regions into its constituent parts or objects. Autonomous segmentation is generally one of the most challenging tasks in digital image processing. A robust segmentation procedure significantly contributes to the successful solution of imaging problems requiring the identification of individual objects.

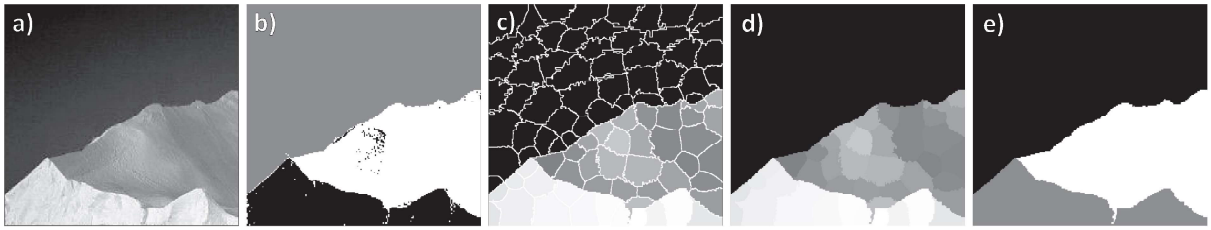


Figure 2.31: Segmentation using k-means (b) vs superpixel algorithm (c,d,e) [32]

Most segmentation algorithms rely on two fundamental properties of image intensity values: discontinuity and similarity. In the first category, the approach involves partitioning an image into regions based on abrupt changes in intensity, such as edges. Approaches in the second category, including thresholding, region growing, region splitting and merging, focus on partitioning an image into regions that are similar according to predefined criteria. Enhanced segmentation performance can be achieved by combining methods from different categories, for example, by integrating edge detection with thresholding. Another approach to image segmentation involves clustering, superpixels, and graph cuts, which is well-suited for extracting the principal regions of an image. Morphology-based image segmentation combines attributes from both discontinuity and similarity techniques. The additional source of information from motion cues can be used to enhance the accuracy and effectiveness of the segmentation process.

Feature extraction almost always follows the output of a segmentation stage, which usually comprises raw pixel data, constituting either the boundary of a region or all the points in the region itself. Feature extraction involves feature detection and feature description. Feature detection involves identifying features within an image, region, or boundary, while feature description assigns quantitative attributes to these detected features. For instance, the determination of the orientation and location of corners in a region boundary constitutes quantitative attributes. Feature processing methods are categorized into three primary groups, depending on their applicability to boundaries, regions, or entire images. Some features can be applied to more than one category. Feature descriptors should exhibit sensitivity as minimally as possible to variations in parameters such as scale, translation, rotation, illumination, and viewpoint. These descriptors are either inherently insensitive to or can be normalized to address variations in one or more of these parameters.

Image pattern classification is the process that assigns a label (e.g., "vehicle") to an object based on its feature descriptors. Methods of image pattern classification range from classical approaches such as minimum-distance, correlation, and Bayes classifiers to more modern approaches implemented using deep neural networks, ideally suited for image processing work.

Techniques for image pattern classification are divided into three principal categories: classification by prototype matching, classification based on an optimal statistical formulation, and classification based on neural networks. The first two approaches are used extensively in applications in which the nature of the data is well understood, leading to an effective pairing of features and classifier design. These approaches often rely on a great deal of engineering to define features and elements of a classifier. Approaches based on neural networks rely less on such knowledge, and lend themselves well to applications in which pattern class characteristics (e.g., features) are learned by the system, rather than being specified a priori by a human designer.

2.3 Deep Learning

This section presents the Artificial Intelligence (AI) paradigm, narrowing down to the foundational concepts of Machine Learning (ML) algorithms for establishing the necessary basis for comprehending Deep Learning (DL) architectural strategies.

2.3.1 Artificial Intelligence Scope

Artificial Intelligence is considered "the new electricity" due to the emergent technological revolution of this century's society. But in fact, the term "artificial intelligence" was coined in 1956 and the creation of the *perceptron* neural network dates to 1957.

It is important to start this subsection by elucidating the hierarchical structure of deep learning by presenting an illustration in Figure 2.32 that delineates their intrinsic relationships, and defining the scientific concepts associated with each term, since those terms (AI, ML, DL) are frequently used interchangeably, especially in mainstream communication channels, which is leading to a loss of the original concepts.

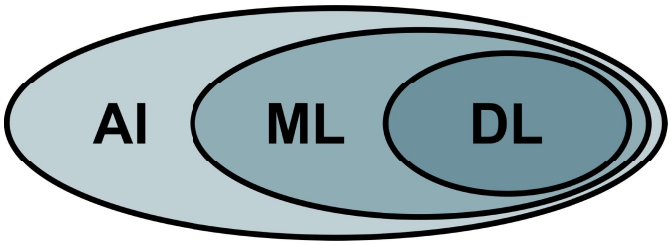


Figure 2.32: Hierarchic structure of deep learning

Artificial Intelligence is the art of engineering intelligent machines and programs. The original concept of the term AI encompasses a broader spectrum of technologies and systems, that just those with learning capabilities. Meaning that every closed-loop control system with memory, observers and actuators has some form of intelligence, by being capable of recognizing patterns and adapt to the environment.

Machine Learning embodies the capability of a system to carry out tasks without being explicit programmed. Instead, it adapts its behavior through data processing, striving to enhance its performance on a given task.

Deep learning is a class of machine learning algorithms that can be distinguished by their deepness, being the deep dimension, the number of layers in a neural network.

Deep learning is the actual field that stands as the revolutionary technology of this century, emerged from the recent increase in computational power and the abundance of available data. Its remarkable results have fueled an increased drive to explore more complex algorithms, enabling the feasibility of deeper networks and yielding remarkable breakthroughs. These advancements span various applications, from speech

recognition to text generation and image segmentation, demonstrating the vast potential and versatility of deep learning across diverse domains and fields.

The deepness of the neural network is merely an outcome driven by what truly distinguishes DL from ML. Traditional ML algorithms are employed to learn patterns from a vector of hand-engineered features extracted from the input. While DL algorithms are end-to-end solutions that are employed directly on the input, learning to extract relevant features, recognize patterns from their combinations, and produce classification outputs. Figure 2.33 illustrates the learning flow of both technologies

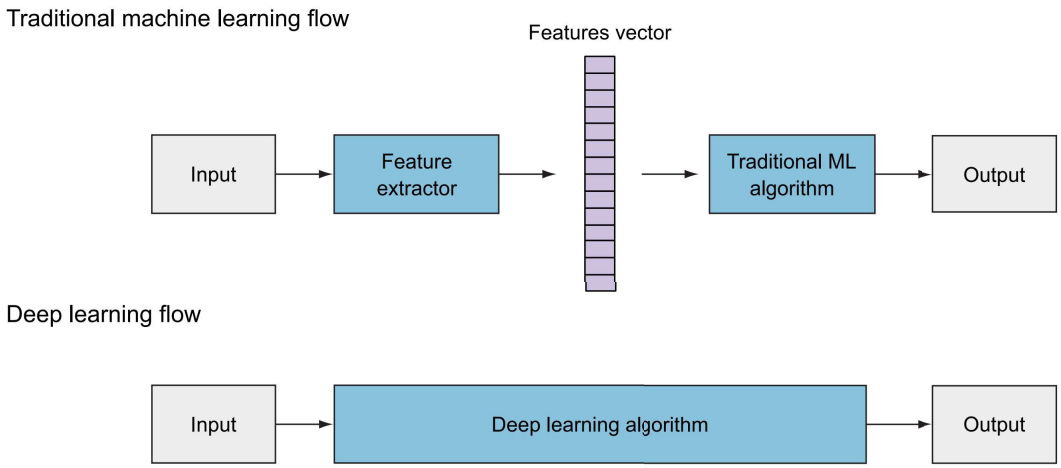


Figure 2.33: Feature extraction difference between ML and DL [45]

To solidify the conceptualization of this idea, it is easy to associate it to a practical case, such as, a task of emotional classification using images of facial expressions.

A machine learning (ML) approach might involve engineering digital image processing functions, such as segmentation and computing region properties, to extract relevant features such as mouth and eyebrow length, curvature, and relative position. Subsequently, based on those feature values, ML algorithms, such as decision trees, random forest, support vector machines, or neural networks, can learn intricate patterns by using all available data as reference for tuning the classification function.

In ML, the initial process, the feature extraction, draws heavily from existing knowledge about the problem. For example, it is widely recognized that features like mouth shape and eyebrow position contribute significantly to defining facial expressions.

However, understanding all relevant combinations of features for classifying a face expression, like the exact mouth curvature ray, that distinguishes between positive and negative emotions is inherently data-driven. Discovering these nuanced decision parameters is more effectively achieved through exploration within a labeled database, a process known as supervised learning.

Various machine learning paradigms address distinct challenges and scenarios, each coupled with the following practical application examples: [46]

Supervised Learning

- **Image Classification:** Identifying objects or scenes within images.
- **Speech Recognition:** Converting spoken language into text.
- **Predictive Analytics:** Forecasting stock prices or weather based on historical data.
- **Medical Diagnosis:** Identifying diseases or conditions from images or patient data.
- **Sentiment Analysis:** Analyzing text to determine emotions or opinions expressed.

Unsupervised Learning

- **Clustering:** Grouping similar customers based on their behavior for targeted marketing.
- **Anomaly Detection:** Identifying unusual patterns in data, such as fraud detection in financial transactions.
- **Dimensionality Reduction:** Reducing the number of features in datasets without losing crucial information.
- **Topic Modeling:** Analyzing large volumes of text to discover hidden topics or themes.
- **Recommendation Systems:** Suggesting movies, products, or content based on user preferences.

Reinforcement Learning

- **Game Playing:** Training AI agents to play games like Chess, or video games.
- **Robotics:** Teaching robots to perform tasks or navigate environments.
- **Autonomous Vehicles:** Developing self-driving cars that learn to make decisions.
- **Resource Management:** Optimizing power usage in data centers or managing supply chains.
- **Personalized Education:** Adaptive systems that tailor content to students' progress.

Self-supervised Learning

- **Image Inpainting:** Filling in missing parts of an image based on its context.
- **Video Prediction:** Predicting future frames in a video sequence.
- **Audio Source Separation:** Separating different sound sources in a recording.
- **Data Augmentation:** Generating synthetic data to improve model robustness.

Each learning paradigm addresses different challenges and scenarios. Supervised learning is prevalent in tasks where labeled data is available, but other paradigms, like unsupervised learning or reinforcement learning, handle scenarios where labeled data is limited or unavailable, or when the objective involves exploring data structures or optimizing behavior in dynamic environments.

2.3.2 Neural Networks

Like most engineered systems that use nature as a source of inspiration, Artificial Neural Networks (ANN) are inspired by the complex network of neuronal connections in the human brain.

A general understanding of a biological neuron is that it receives electric signals from other neurons to which it is connected, and when activated, produces an output signal that is successively connected to other neurons as well. Applying this analogy to artificial neurons involves the concept of receiving input information from various connections and generating an output value based on its activation state.

However, it is important to note that this analogy, while widely accepted, does not capture the complete complexity of a single biological neuron, which remains a subject of ongoing research in neuroscience [47].

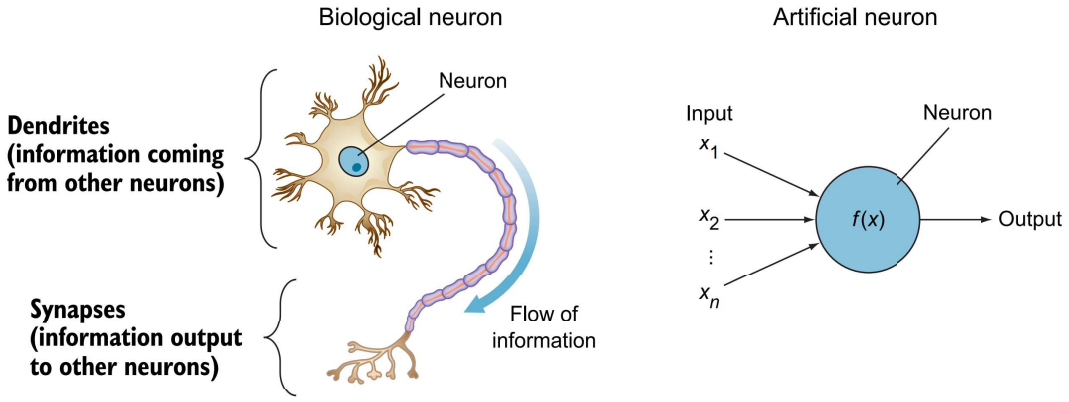


Figure 2.34: Similarities between biological neurons and artificial systems [45]

Formed by a set of neurons organized in a set of layers, a neural network (NN) starts with an input layer, that receives a single entity of data structured as a vector, where each neuron represents a unique feature of this entity, and ends with an output layer that might hold a set of characteristics about the processed entity.

If the entity is a photovoltaic module, the features may be type, location, age, and other features that could be related to its efficiency. In this case, the NN output layer is a single neuron that predicts the energy conversion efficiency factor based on its known features.

When referring to images, each pixel can be considered as a feature. Each neuron of the input layer assumes a value corresponding to each pixel of the addressed image.

In an Optical Character Recognition (OCR) algorithm, each neuron of the NN output layer should be a character, making the output size match the number of characters considered. The value in each output neuron represents the confidence of the prediction, ranging from zero to one. The highest value over all the output vector has an index which is related to the character identification.

Between those extreme layers reside a set of hidden layers. Figure 2.35 illustrates the general architecture of a standard neural network where all neurons or nodes have one connection with every other node in the neighbor's layers.

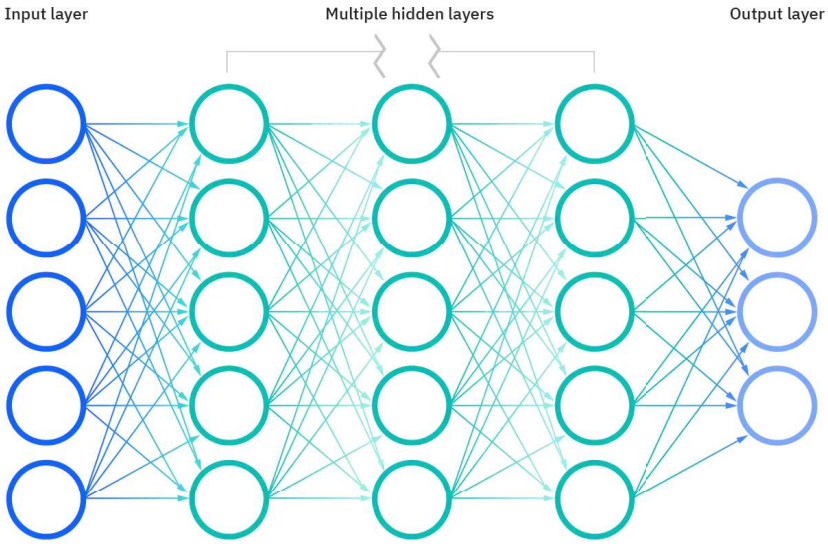


Figure 2.35: Representation of a standard neural network

Before exploring the forward propagation concept, it might be useful to look at a simple study case, to get an intuition about the "magic" surrounding a neural network.

It is important to clarify what a single neuron does, starting by analyzing the smallest possible NN, with just one input and one output.

Plotting the data of a PV module, allows to observe a relation just like the one illustrated in Figure 2.36, where the Age is the input and the Efficiency is the output.

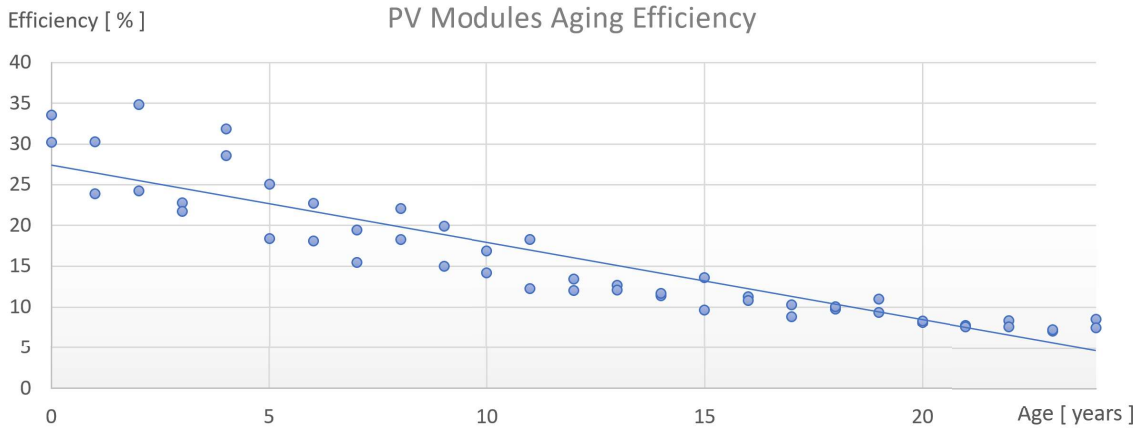


Figure 2.36: Linear regression study case

It can be easily recognized that this trained network performs exactly as a linear regression. Meaning that a neuron is just a linear function, with parameters that have been calculated by minimizing the error between the data and the function output over all the training data set.

2.3.2.1 Forward Propagation

To understand what information all those connections between neurons carry, a minimal NN, also known as perceptron, with a single node, a single output, and few inputs is introduced in Figure 2.37 to enable visual intuition.

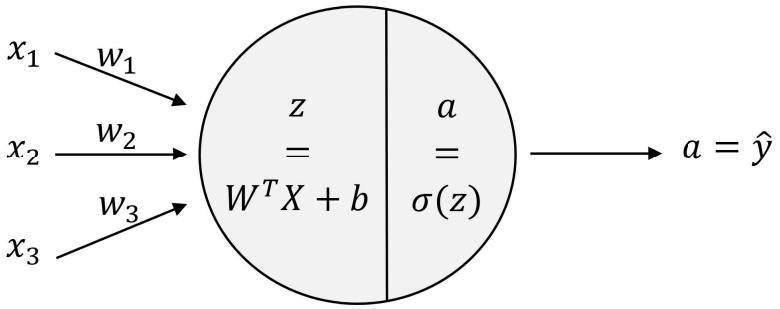


Figure 2.37: Representation of a single neuron network

The features of an entity are the inputs of the NN, in this specific case $\boldsymbol{x_1}$, $\boldsymbol{x_2}$, and $\boldsymbol{x_3}$. The output of the NN is $\hat{\boldsymbol{y}}$ that is described as the function predicted value. $\boldsymbol{y}$ is the value of labeled data to use for training the NN parameters. The connection weight $\boldsymbol{w_i}$ is a coefficient for $\boldsymbol{x_i}$, making part of a column matrix $\boldsymbol{W}$. The weighted sum is the product of the transposed weight matrix $\boldsymbol{W^T}$ by the input matrix $\boldsymbol{X}$. The input of the activation function $\boldsymbol{a}$ is $\boldsymbol{z}$ that shift the computed weighted sum by a bias $\boldsymbol{b}$, and the output of the activation function $\boldsymbol{a}$ in the last layer is equal to the output of the NN $\hat{\boldsymbol{y}}$.

A more general notation (Equation 2.1) describes $\boldsymbol{z_{i,j}^{[l]}}$ as the weighted sum of the activation of the previous layer, shifted by a bias, being $\boldsymbol{l}$ the current layer, $\boldsymbol{j}$ a node from the current layer, $\boldsymbol{k}$ a node from the previous layer, and $\boldsymbol{i}$ a current training example.

$$z_{j,i}^{[l]} = \sum_{k=1}^{n^{[l-1]}} w_{j,k}^{[l]} a_{k,i}^{[l-1]} + b_j^{[l]} \tag{2.1}$$

Using the previous neural network architecture (Figure 2.37) to obtain a PV module efficiency prediction model with features like technology type, geographic location, and operating age will lead to a four-dimensional linear function.

However, this model presents a challenge due to its linear function in a four-dimensional space. Despite these features being indirectly linked to the output, it's doubtful that this simplistic model could accurately predict outcomes.

To address this limitation, combining these distinct domains of features might yield more relevant relationships to the output. For instance, merging technology type with the production year could establish an initial efficiency point. Similarly, combining localization and operational age might help modeling the degradation level, that is closely tied to efficiency decline.

Since the identified issue is the significant level of simplicity, the solution involves exchange simplicity for complexity. Introducing additional layers is a means to expand the function's complexity. Adding just another layer allows obtaining the architecture of a neural network presented in Figure 2.38.

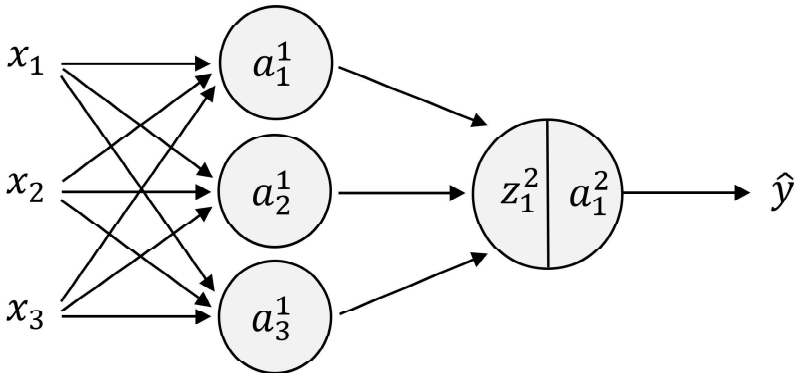
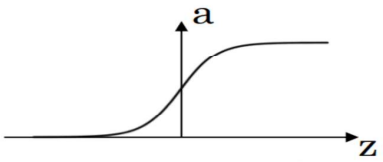
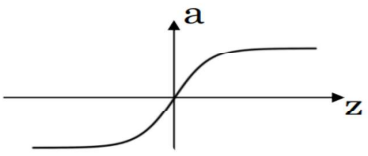
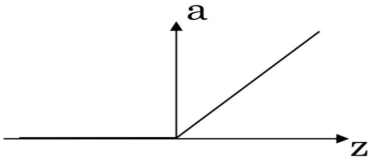
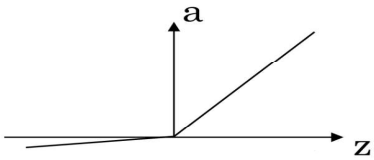


Figure 2.38: Representation of a two layers neural network

The rise in complexity is translated in practice by increasing the number of steps in a piecewise function. This hybrid function is defined by intervals across the domains of the different dimensions. Those intervals are achieved by the activation functions, that are explored in Table 2.5.

Table 2.5: Activation Functions

Function	Expression	Codomain	Illustration
sigmoid	$\frac{1}{1+e^{-z}}$	$]0;1[$	
tanh	$\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$]-1;1[$	
ReLU	$\begin{cases} z, \text{ if } z > 0 \\ 0, \text{ if } z \leq 0 \end{cases}$	$[0;\infty[$	
LeakyReLU	$\begin{cases} z, \text{ if } z > 0 \\ 0.01z, \text{ if } z \leq 0 \end{cases}$	$]-\infty;\infty[$	

Let's consider now a two-dimensional function, with just one input, one output, and parameters corresponding to one hidden layer with tree neurons, equivalent to a function composed by the sum of tree functions. The result is a linear relation between the input and the output if all activation functions are linear, as shown in Figure 2.39.

Thus, it is necessary to employ non-linear activation functions to achieve a complex solution that properly modulates a system of interest.

The Figure 2.40 and the Figure 2.41 shows examples of the stated complexity, achieved by the combination of simple activation functions.

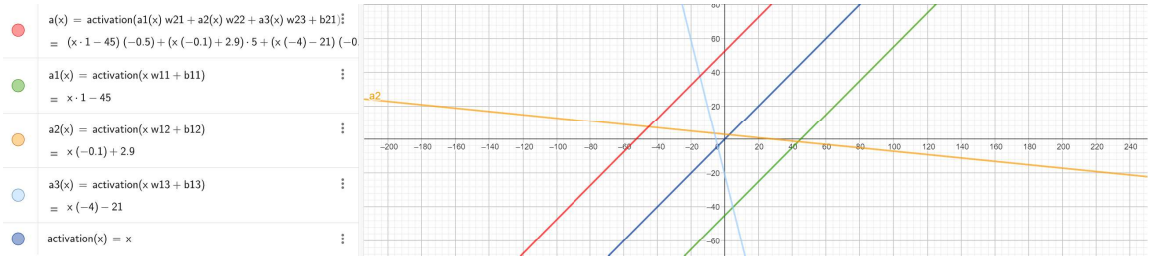


Figure 2.39: Function composition example using Linear Activation [48]

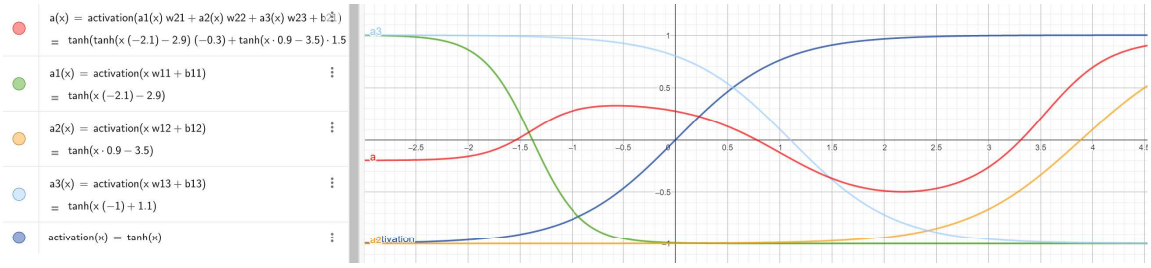


Figure 2.40: Function composition example using Hyperbolic Tangent Activation [48]

Figure 2.41: Function composition example using ReLU Activation [48]

The Rectified Linear Unity (ReLU) activation function has had positive surprising empirical results in performance and consequently in precision by allowing increase the complexity. A motivation for that outcome is the low computational effort required to calculate that function. Computing ReLU is perform a neutral multiplication after a logic comparison between floats, in contrast to an iterative method which uses rotations to calculate hyperbolic and trigonometric functions, a recognized hardware-efficient algorithm named CORDIC (COordinate Rotation DIgital Computer) [49].

2.3.2.2 Back-Propagation

During forward propagation, the input data is passed through the neural network's layers, undergoing transformations through various activation functions and weights. The information flows forward, layer by layer, until it reaches the output layer, which generates a prediction. In contrast, backpropagation moves in the reverse direction, from the output layer back to the input layer. It starts with the calculated difference (or error) between the predicted output and the actual output (the loss function). This error is then propagated backward through the network, and gradients are calculated at each layer with respect to the loss. These gradients are used to adjust the network's parameters (weights and biases) to minimize the error in subsequent iterations [50].

The computed Loss Function depends on the problem's scope, as outlined in Table 2.6.

Table 2.6: Loss Functions

Problem	Function	Expression
Regression	Mean Squared Error (MSE)	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Binary Classification	Binary Cross-Entropy	$-\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$
Multi Classification	Categorical Cross-Entropy	$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{M} y_{ij} \cdot \log(\hat{y}_{ij})$

Gradient Descent is an optimization algorithm that computes the rate of change (or slope) of the loss function concerning each weight and bias in the network, by using partial derivatives that indicate the direction and magnitude of the steep of the loss function concerning each parameter. These derivatives or gradients guide the updates made in the parameters to reduce the loss iteratively, aiming to find the optimal set of weights and biases that minimize the overall loss, as illustrated in Figure 2.42.

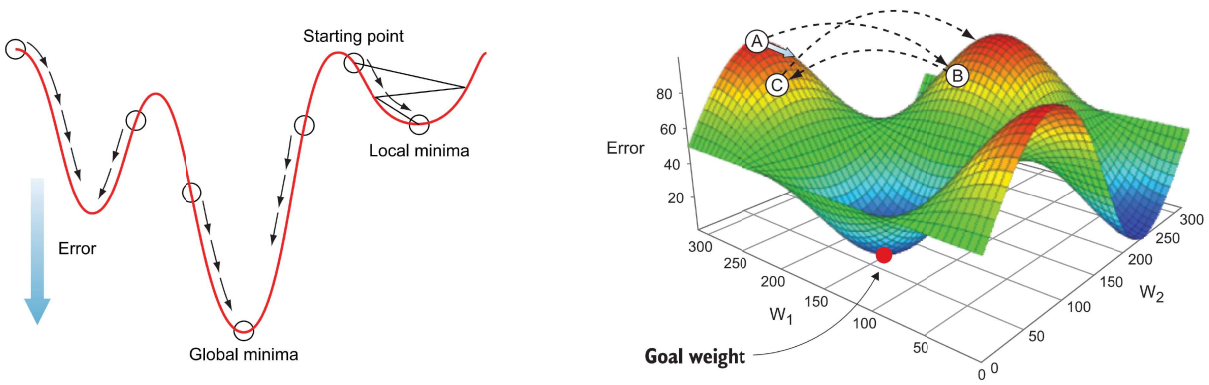


Figure 2.42: Representation of gradient descent iterations [45]

2.3.3 Deep Vision

Referring to computer vision systems employing deep learning methodologies, Deep Vision (DV) encompasses three primary problem domains:

- **Image Classification** involves using an image as the system input and generating an output prediction value associated with a class label.
- **Object Detection** expands on this by including not only class label predictions but also providing two pairs of values for bounding box localization and dimensions for each identified object within an image.
- **Style Transfer** takes a pair of images as input and generates an output that aims to minimize the discrepancy between their features, as Figure 2.43 illustrates.

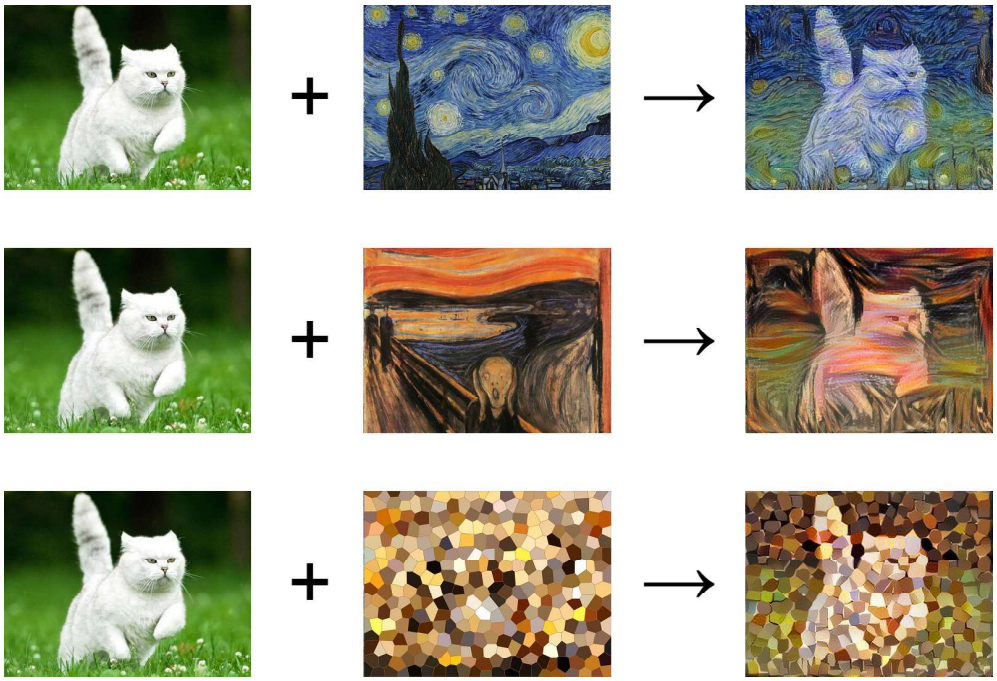


Figure 2.43: Results of using different style references on the same original image [51]

Moreover, the fusion of natural language processing (NLP) and deep vision (DV) has led to the development of text-to-image generation models. These models empower architectures like Generative Adversarial Networks (GANs) or Variational Auto-Encoders (VAEs) to create realistic synthetic images based on textual descriptions.

A notable characteristic of deep vision lies in the input data type. Matrices of decompressed images, can reach colossal dimensions. For instance, processing a 1-Megapixel (MP) RGB image involves handling 3,000,000 values, similarly to decoding a password pattern with 3,000,000 symbols through fully connected layers. Additionally, images retain spatial relations that are not effectively captured in a vector format. To address these properties, convolutional layers are the main choice to effectively interpret 2D patterns, extracting important features from raw data.

2.3.3.1 Convolution

In mathematics, convolution is an operation that uses a function return as argument of other function. In the Convolutional Neural Networks (CNNs) context, a convolution operation involves sliding a filter or kernel (a small matrix) over the input image. At each position, is computed an element-wise multiplication between the filter values and the overlapping pixel values in the input image. These products are summed to produce a single value in the output, the convolved image, known as feature map. Figure 2.44 describes and exemplifies the convolution operation [45].

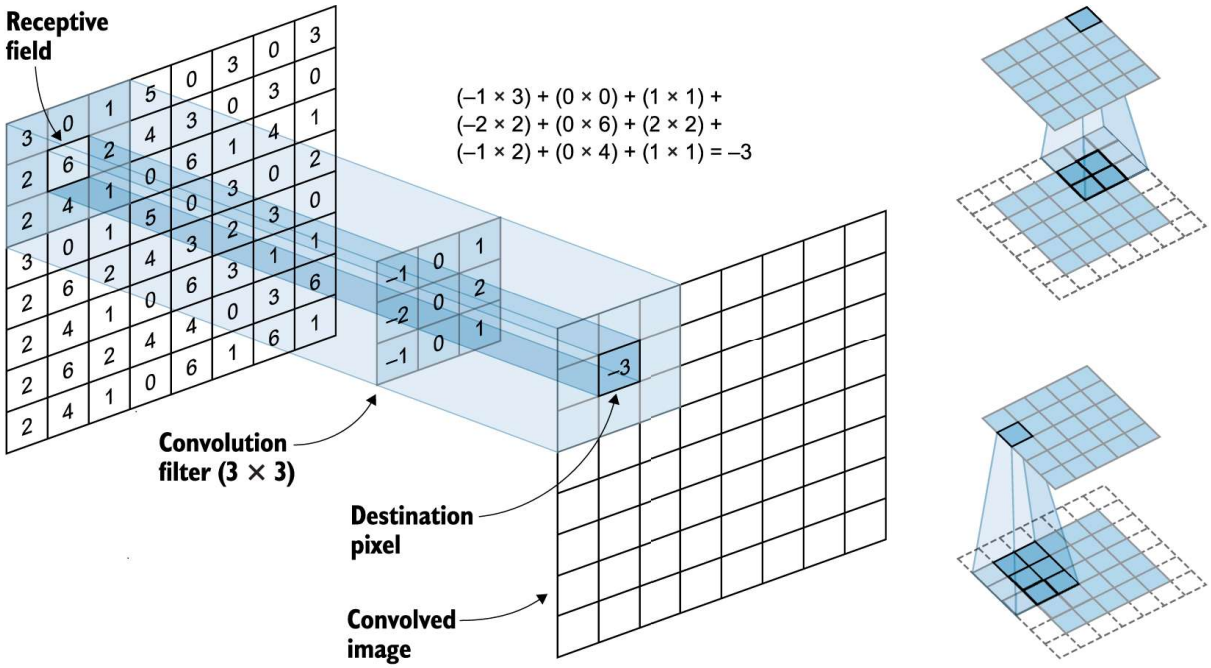


Figure 2.44: Convolution operation illustrative description [45]

Padding is used to maintain the same size in the output. The pixels added to the input can assume various values, such as zero, the mean of the entire input image, the value of the next pixel, reflect the pixels perpendicular to the edge, or even use other frequency domain functions. When no padding is performed, the output feature map dimension is reduced to $(N_1 - F_1 + 1) \times (N_2 - F_2 + 1)$, where $N_1 \times N_2$ is the input size and $F_1 \times F_2$ is the filter size.

The Stride value represents the step size of the sliding operation during convolution. Pooling involves using a convolutional-like operation with a kernel that produces either maximum or average values, often employed with a certain stride to decrease the size of the feature map. Notably, the necessity of Pooling layers has been questioned lately as striding in convolutional operations can yield similar practical outcomes [52].

The values within a kernel dictate the resulting feature map. Certain kernels, when their values sum to one, have the ability to smooth or sharpen an image. Kernels with a sum of zero can eliminate the background and identify either vertical or horizontal edges, or a combination of these directions based on the kernel's dimensions and values, as illustrated in Figure 2.45.

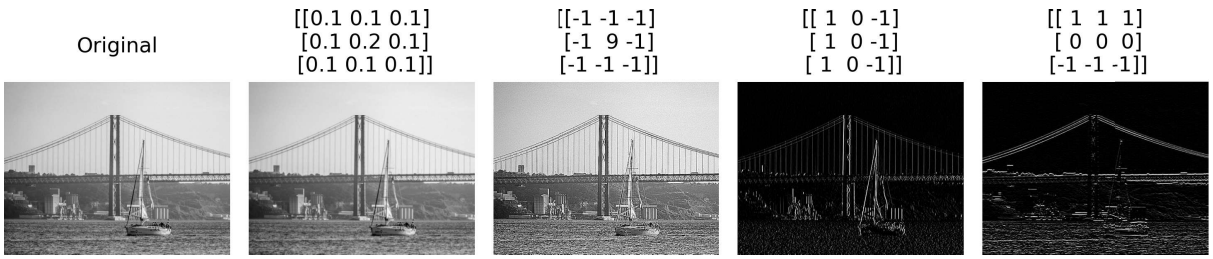


Figure 2.45: Feature maps of different kernel values

The kernels in convolutional layers are the parameters that are learned during the training process. These kernels start with random values and get adjusted throughout training using optimization algorithms like gradient descent, aiming to minimize the error between predicted and actual outputs.

The objective is for the network to learn filters that can extract meaningful features from the input data, like edges, textures, shapes, or higher-level features as the layers progress. These learned filters become specialized in recognizing specific patterns within the input data, which aid in classification, detection, or other tasks.

Through the arrangement of multiple convolutional layers, CNNs create a hierarchical representation of features. Early layers detect simple features like edges and gradients, while deeper layers detect complex patterns and high-level representations. For instance, the combination of edge patterns could form simple shapes like corners or circles, and their combinations might create more complex polygons. Overlapping these polygons can produce figures with various textures, progressing from basic objects to increasingly intricate and comprehensive ones.

Figure 2.46 shows the learned kernel values represented in normalized grayscale, demonstrating the convolutional filters' evolution after several learning iterations.

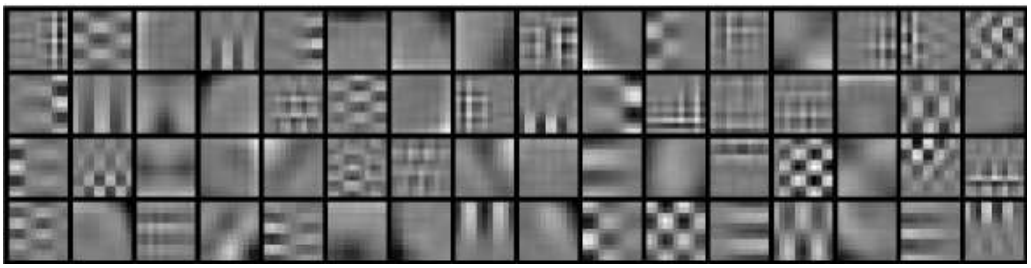


Figure 2.46: Learned convolutional kernels in a CNN [53]

2.3.3.2 Classic Networks

Every CNN model has two main blocks: the first is the feature extractor, comprising convolutional layers; and the last is the classifier, containing dense layers. Typically, as the network progresses through the layers of a CNN, the spatial dimensions tend to decrease, while the depth may increase, starting with a high dimension matrix and ending up with a reduced vector. Figure 2.50 presents an overview of CNN's evolution.

All started in 1998, when Lecun *et al.*introduced a pioneering CNN called LeNet-5 [54], composed of five weight layers, three convolutional layers and two fully connected layers.

In 2012, AlexNet, developed by Krizhevsky *et al.*, marked a significant leap in computer vision, after wining an image classification competition, demonstrating its capability to classify 1.2 million high-resolution images into 1,000 different classes from the ImageNet dataset [55].

VGG16 [56], developed in 2014 by the Visual Geometry Group at Oxford University, emerged in 2014, with an architecture like LeNet5 and AlexNet, only deeper, housing 138 million parameters across its 13 convolutional layers and 3 dense layers, that are presented by the Figure 2.47 illustration.

2014 was remembered in the deep learning history also by a groundbreaking architecture, codenamed Inception [57],introduced by group of researchers at Google, that incorporating a novel element named as the inception module, presented in Figure 2.48, that employs diverse kernel sizes within a single layer, utilizing parallel convolutions and pooling to capture multiscale features efficiently. This innovation enabled a network 22 layers deep attain only 13 million trainable weights, deeper than VGGNet and with 12 times fewer parameters, while achieving improved accuracy.

ResNet [58], introduced by Microsoft in 2015, further revolutionized deep learning by introducing residual connections, in residual blocks as illustrated in Figure 2.49, that tackled the vanishing gradient problem in very deep networks, allowing for training of even deeper architectures up to hundreds of layers.

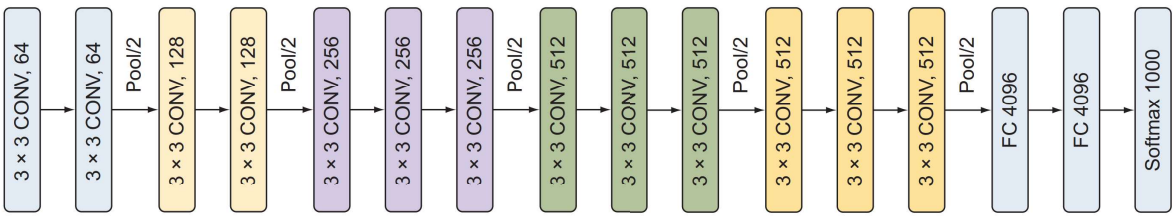


Figure 2.47: VGG16 architecture overview [45]

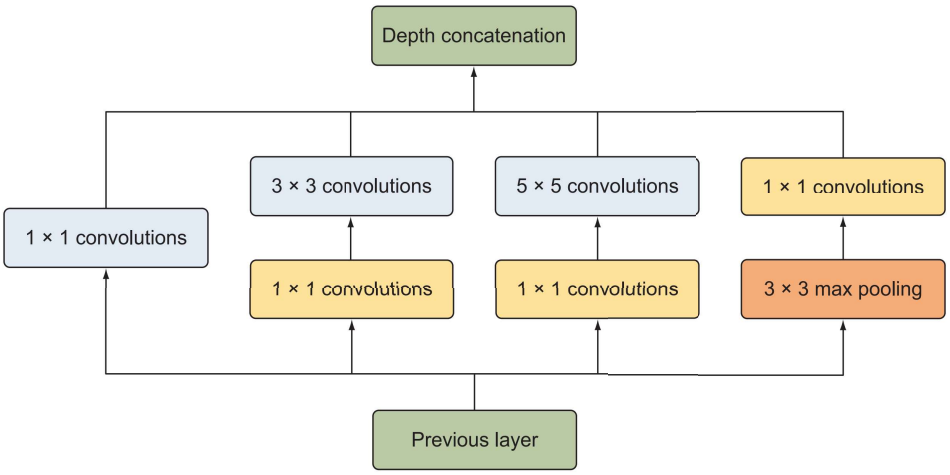


Figure 2.48: Inception block architecture overview [45]

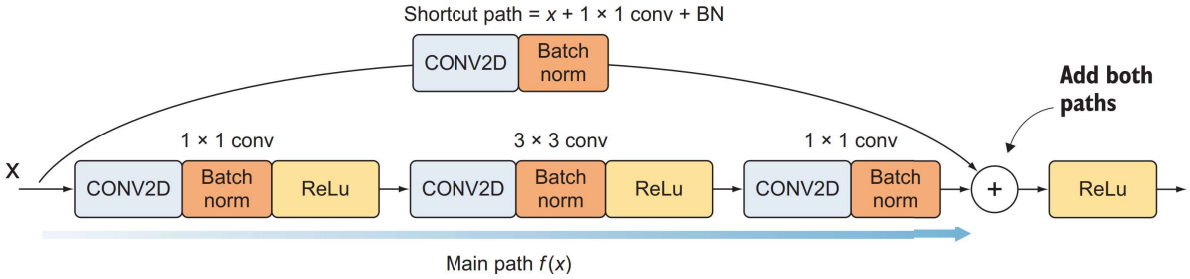


Figure 2.49: ResNet block architecture overview [45]

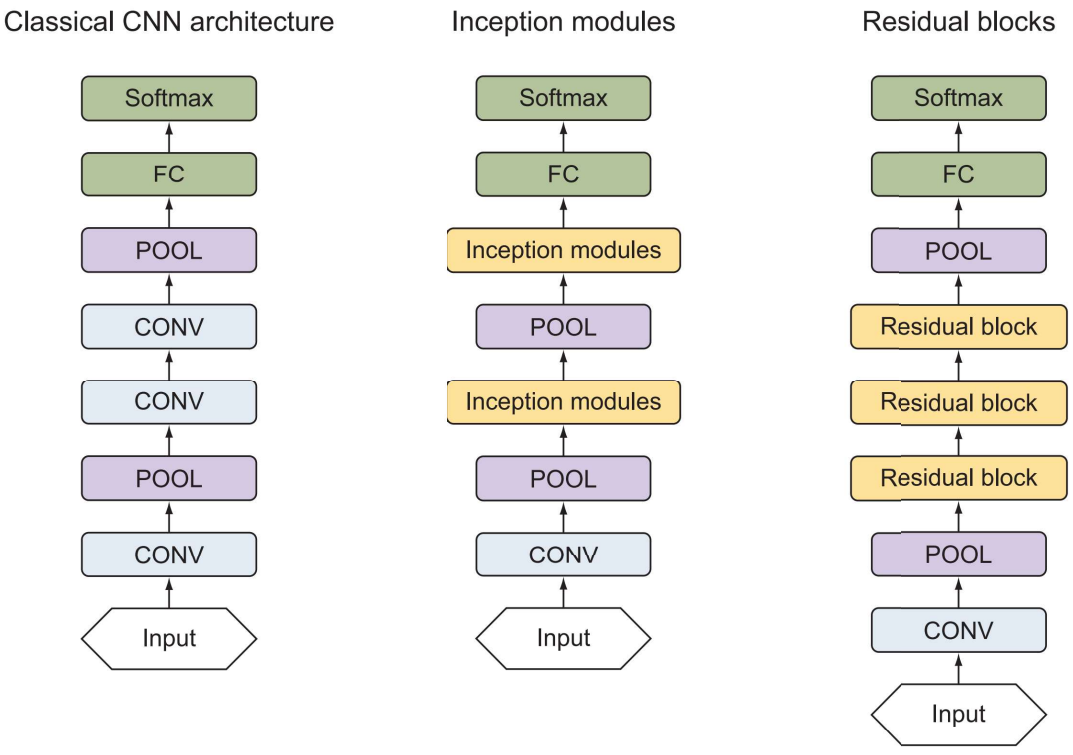


Figure 2.50: CNN architecture evolution [45]

2.3.3.3 Object Detection

Image Classification is limited to identifying a single object for each image. Identifying multiple objects in one image is known as Object Detection, and involves localizing and classifying each object by predicting bounding boxes and their respective classes.

The framework of object detection systems employing DL follows three main steps, illustrated in Figure 2.51 and described as follows.

Region Proposal identifies potential object-containing regions within an image by distinguishing them from the background based on the density of features.

Network Predictions consists on performing the analysis of the identified Regions of Interest (ROIs), employing CNN features extractors to generate predictions of the object classes and the Bounding Box (BB) coordinates (x,y,w,h).

Non-Maximum Suppression is a technique used to eliminate multiple detections of the same object, retaining only the most confident and accurate one. After discarding the predictions below a *confidence threshold*, elect the highest score BB, evaluate the BB's overlapping, or intersection, and suppress redundant predictions for the same class.

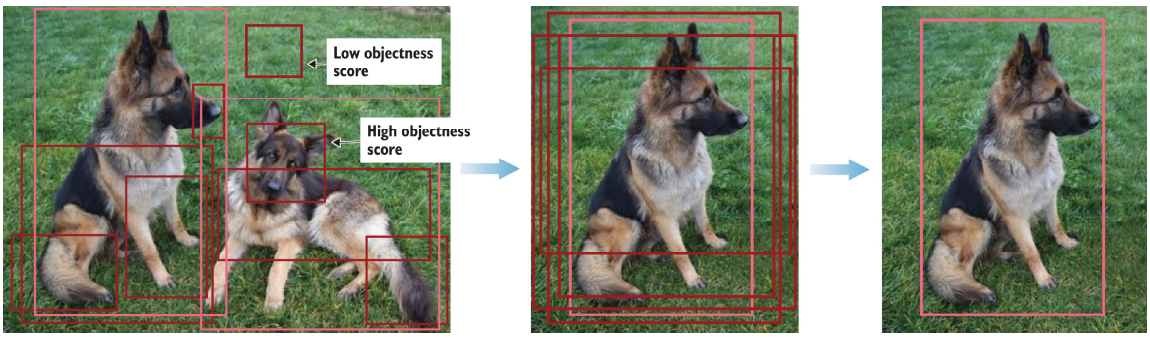


Figure 2.51: Object Detection Steps [45]

The main evaluation metrics for object detection, such as mean Average Precision (mAP) and mean Average Recall (mAR), are associated with IoU threshold limits.

Intersection over Union (IoU) quantifies the relative overlapping between the Ground Truth (GT) and the object BB prediction, as shown in Figure 2.52.

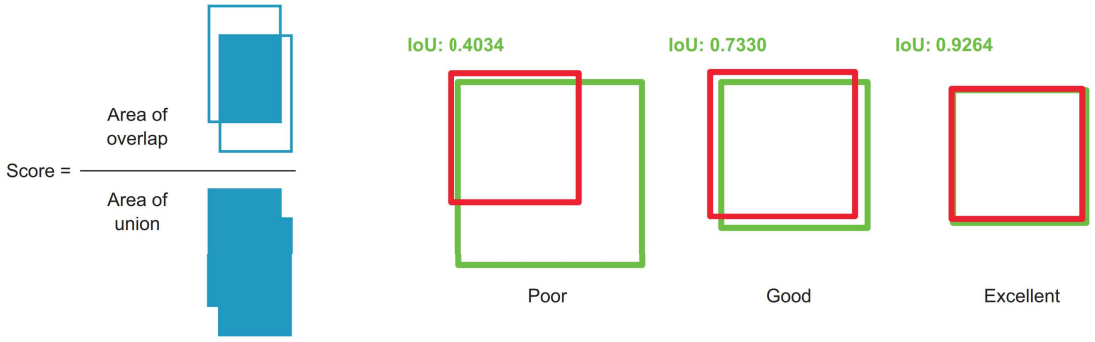


Figure 2.52: Intersection over Union [45]

Object detection algorithms have evolved over the last years as illustrated in Figure 2.53. The three most popular network families are the R-CNN, the YOLO, and the SSD.

All models within the R-CNN family are region-based, operating as two-stage detectors. These employ a process of proposing a set of Region of Interests (RoIs) using methods like selective search or a Region Proposal Network (RPN). Following this, the classifier exclusively processes the region candidates.

One-stage detectors take a different approach. They skip the region proposal stage and run detection directly over a dense sampling of possible locations. This approach is faster and simpler but can potentially drag down performance a bit.

In general, single-stage detectors tend to be less accurate than two-stage detectors but are significantly faster, making them suitable for real-time object detection tasks.

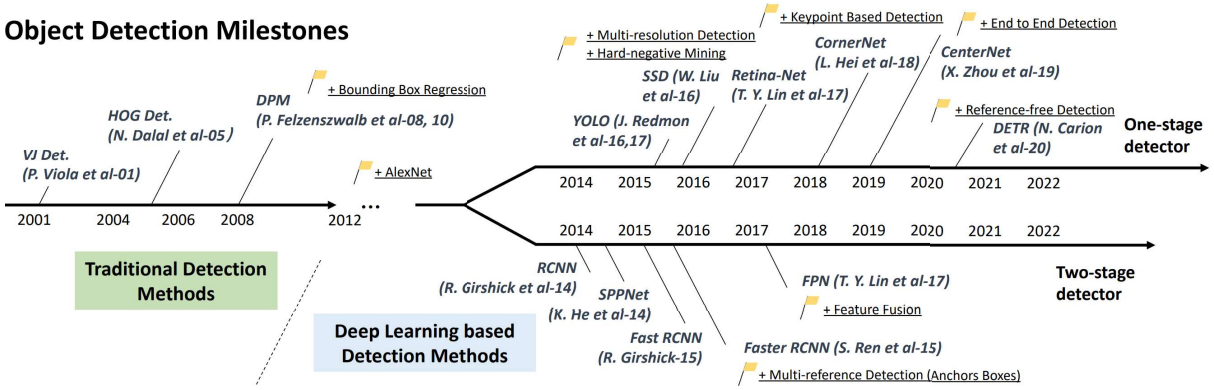


Figure 2.53: Timeline for object detection milestones [59]

The Faster R-CNN model [60], represented in Figure 2.54, tackled the computational bottleneck of the Fast R-CNN by replacing a traditional region proposal algorithm for a Region Proposal Network (RPN) that shares the convolutional features for better inference performance. The images feeding the CNN originate features, such as edges, shapes and object parts, that are encoded in feature maps and processed by the RPN, which tests different box shapes through these maps using larger sliding steps, resulting in a set of anchor boxes, each with a respective score related to the probability of an object's existence.

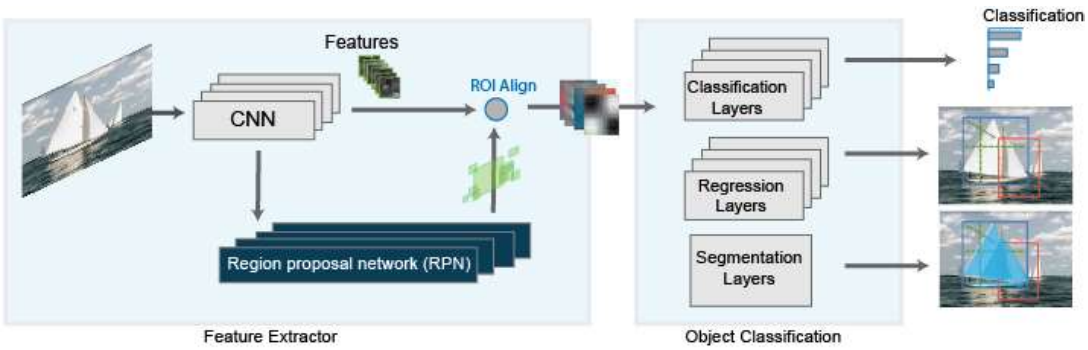


Figure 2.54: Faster R-CNN architecture overview [61]

Single Shot Detector (SSD) operates by using a single neural network, as shown in Figure 2.55, to generate predictions directly from the full image. It divides the image into several grids of different aspect ratios and scales and applies convolutional filters to these grids. These filters predict both the class scores for each object category and the bounding box offsets for multiple default boxes at each grid cell.

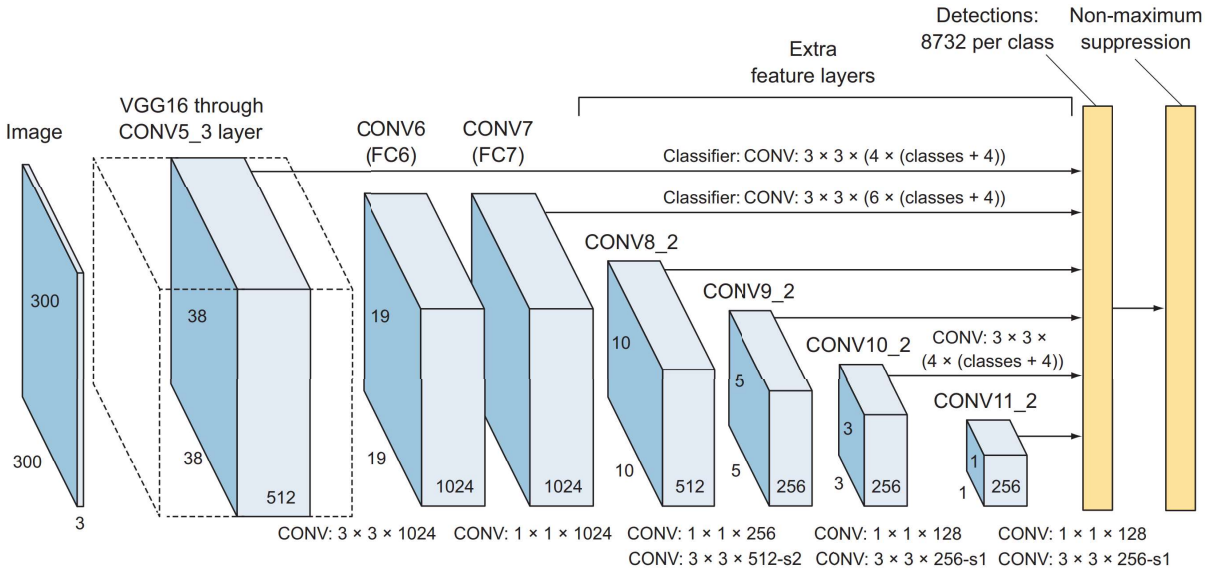


Figure 2.55: SSD architecture overview [45]

YOLO, on the other hand, divides the input image into a grid and operates by processing the image once through a neural network. It operates on the entire image at once and generates predictions using a single neural network pass, which allows it to be extremely fast. YOLO predicts bounding boxes and class probabilities for a fixed number of predefined anchor boxes per grid cell as illustrated in Figure 2.56.

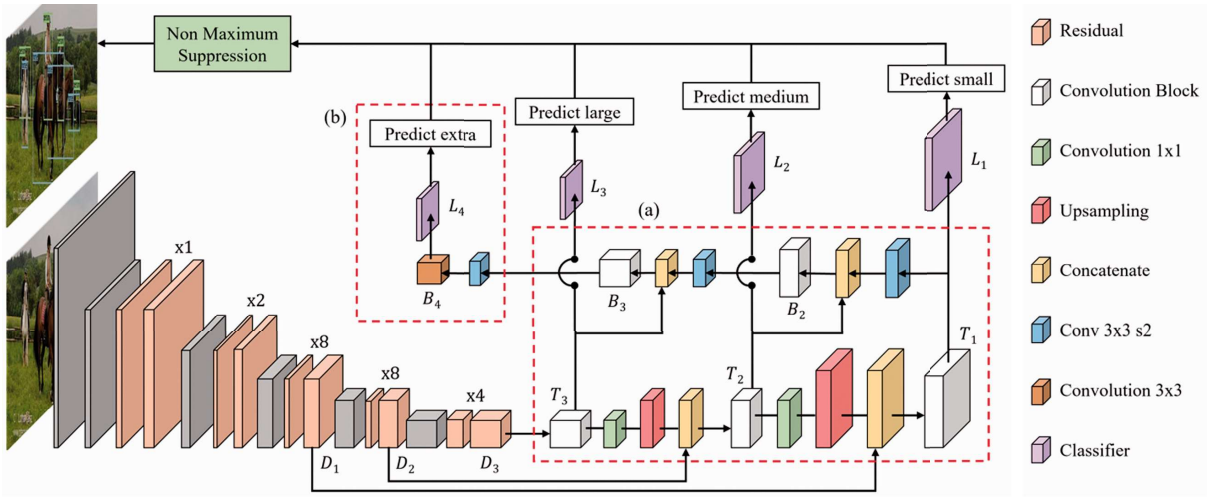


Figure 2.56: YOLOv3 architecture overview [62]

To achieve faster inference times, models might utilize backbone architectures that have fewer parameters or reduce the overall complexity of the network.

By simplifying the backbone architecture or using shallower networks, the model can process information more quickly. However, this reduction in complexity might lead to a loss in accuracy as the model might struggle to capture intricate patterns or subtle details in the images [63]. This trade-off is often reported, as shown in Figure 2.57.

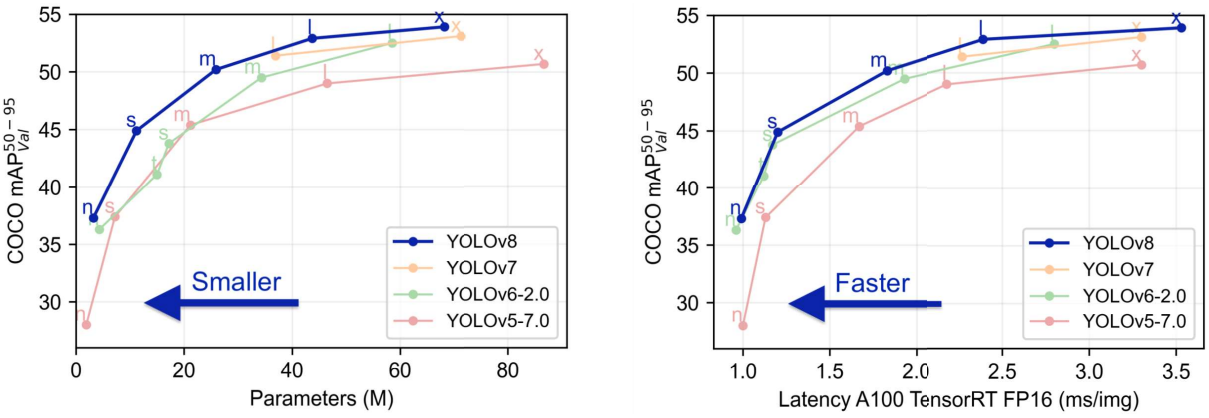


Figure 2.57: Accuracy-Speed trade-off on YOLO family [64]

Large datasets have evolved over the years, becoming more challenging due to increased complexity, diversity, detailed annotations, and benchmark metrics.

This evolution, as illustrated in Figure 2.58, has been an important stimulus for the emergence of new and improved architectures.

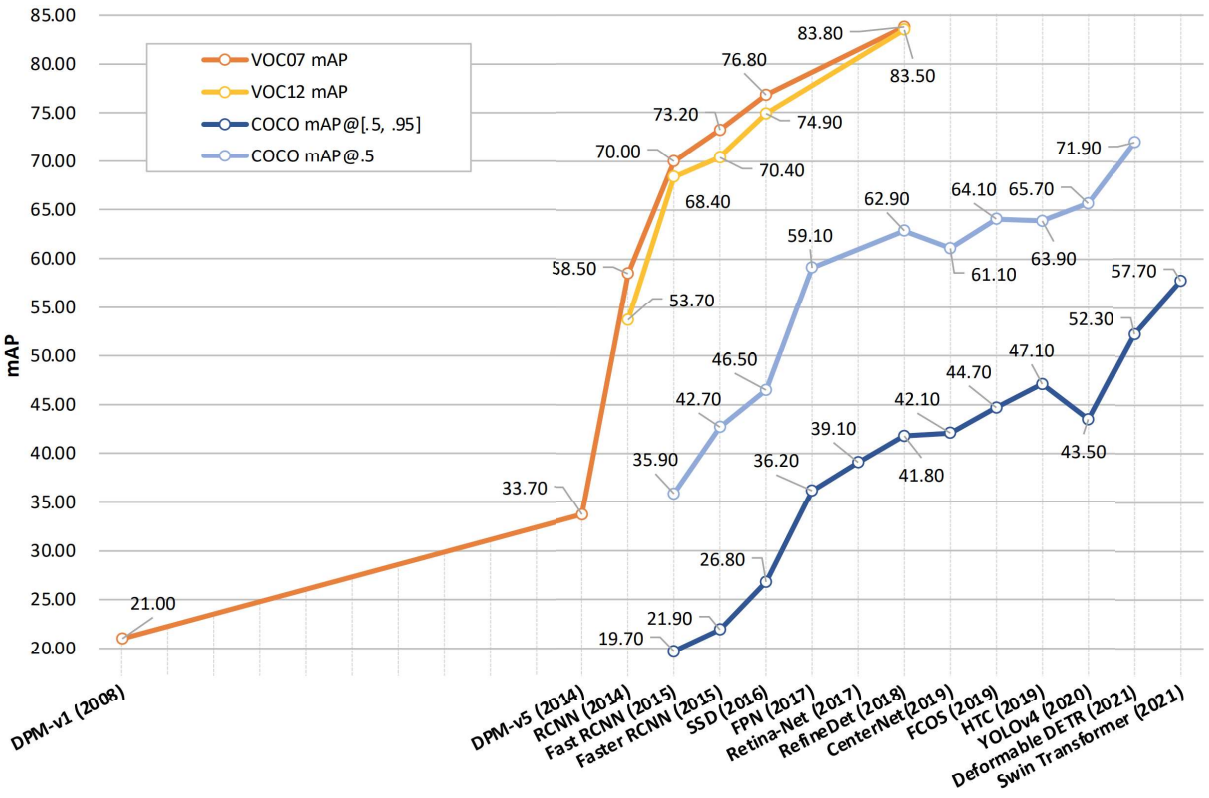


Figure 2.58: Accuracy improvement of main object detection models over time [59]

2.3.4 Optimization Strategies

In this section, various techniques used to improve the model's generalization and robustness are explored. From tuning hyperparameters, applying regularization techniques and methods and selecting the most effective evaluation metrics.

Identifying the appropriate architecture for the problem under consideration is the initial challenge. Following this, the learning process is analyzed, and depending on the outcomes, additional techniques to handle both the data and the architecture, are iteratively employed.

When dealing to hyperparameter tuning, certain rules of thumb can steer the project course in the right direction. The main source of feedback arises from how the model fits to the data.

2.3.4.1 Hyper Parameters

Neural network parameters are the learnable coefficients within the network, while hyperparameters operate at a higher level, influencing the learning process.

These hyperparameters can be categorized into these three main groups:

- **Network architecture**
  - Number of hidden layers (network depth)
  - Number of neurons in each layer (layer width)
  - Activation type
- **Learning and optimization**
  - Learning rate and decay schedule
  - Mini-batch size
  - Optimization algorithms
  - Number of training iterations or epochs (and early stopping criteria)
- **Regularization techniques**
  - Loss penalties
  - Dropout layers
  - Data augmentation

Tuning hyperparameters is an iterative process usually conducted through trial and error. This process can be automated by higher-level algorithms that train various models with different hyperparameters. This approach, known as AutoML, traditionally implements techniques like grid or random search. An emerging and optimized method within AutoML is Bayesian Optimization, which efficiently explores the hyperparameter space using Bayesian reasoning, often requiring fewer iterations compared to traditional methods [65].

2.3.4.2 Model Fitting

A model can be underfitting when it fails to capture the main features of the dataset used, or overfitting when it captures every tiny pattern, including noise, and thus fails to generalize to new data. Figure 2.59 demonstrates that concept [66].

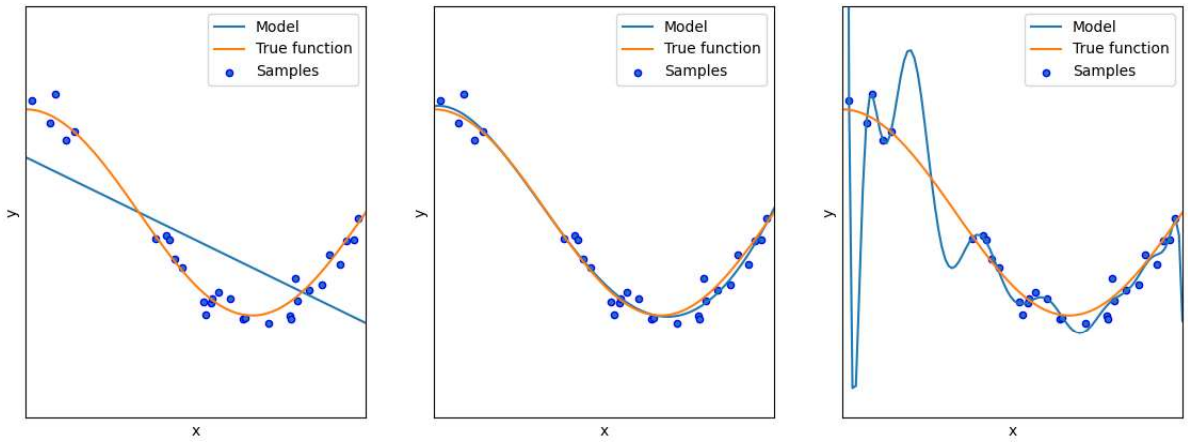


Figure 2.59: Under-fitting, Optimal-fitting, and Over-fitting examples [67]

The most practical method used for evaluating a model's fitness involves comparing the accuracy on both the training and validation splits of the dataset. Meaning that a portion of the data is used to train the model, while another portion is reserved to validate its performance. Table 2.7 covers the main symptoms and solutions.

Table 2.7: Symptoms and Solutions of Model Fitting Issues

Issue	Symptoms	Solutions
Under-fitting	High training error High validation error	Increase model complexity Reduce regularization Reduce data outliers
Over-fitting	Low training error High validation error	Reduce model complexity Increase regularization Increase data augmentation
Optimal-fitting	Low training error Low validation error	Test with new data Fine-tune hyperparameters Optimize computing efficiency

Regularization techniques are essential for ensuring that the model's predictions are generalized. Normalization layers provide significance equality to the features. Dropout, another regularization technique, randomly discarding some trainable parameters during each backpropagation step. This prevents certain neurons from dominating the predictions made by the model, leading to a more balanced learning process [68].

Another recognized technique involves adding penalties to the loss function. These penalties increase with high divergence in weight values across the neural network. Consequently, every neuron contributes in some measure to the model's prediction, reinforcing the principle that the complexity of the function achieved is intricately linked to the dimensions of the neural network [69].

To build a robust and generalized model, having an abundance of diversified data is crucial. When facing limitations in data availability, employing **Data Augmentation** is a solution. This includes transforming the images slightly enough to maintain its features accurate, but different enough to expand the data domain.

When managing both data and computational limitations, **Feature Selection** proves to be highly effective. This can be attained by cropping images or defining Regions of Interest (ROIs) to eliminate irrelevant features.

Another valuable tactic is using previously trained models with similar data, taking advantage of advanced architectures to generate valuable feature maps, with a fraction of the employed computational resources, requiring only the fitting of the classifier block, a strategy known as **Transfer Learning**.

2.3.4.3 Evaluation Metrics

The final step in the iterative process of developing a DL model involves evaluating its performance using appropriate metrics [70].

One commonly used metric is accuracy, calculated using Equation 2.2:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.2}$$

However, accuracy can be misleading, especially when there is an imbalance among classes. For example, achieving 90% accuracy when 90% of the data belongs to a single class may not reflect good performance. If the model consistently predicts a single output, regardless of the input, will also achieve 90% accuracy.

Deeper insights are derived from metrics that collectively form the confusion matrix, as outlined in Table 2.8 and Figure 2.60.

Table 2.8: Confusion Matrix Metrics

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 2.60: Multi-class confuson matrix

Precision (Equation 2.3) evaluates specificity, while Recall (Equation 2.4) assesses sensitivity. F1-Score (Equation 2.5) combines both measures using the harmonic mean.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.3}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.4}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.5}$$

Some applications prioritize reducing the false positive rate, while others focus on minimizing the false negative rate. The trade-off between these objectives can be assessed by examining the ROC curve, presented in Figure 2.61, and setting the threshold to an ideal value. Ultimately, the Area Under the ROC Curve (AUC) serves as a reliable measure of the model's optimal fitting[71].

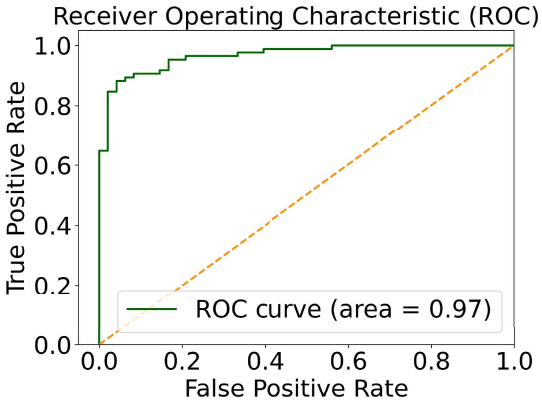


Figure 2.61: ROC curve representation

3 Framework and Design Methodologies

In this chapter, the used framework and tools are introduced. Namely, the software environment, the coding notebook, and common useful libraries and APIs, providing the necessary references for reproducing the forthcoming results of this project.

3.1 Environment

The current trend in machine learning projects gravitates towards Jupyter Notebooks, known for their interactive and document-like interface, seamlessly integrating code, visualizations, and explanatory text. These notebooks are available in both local and cloud-based environments. Figure 3.1 illustrate the popular platform environments.

Cloud-based platforms such as Colab, Kaggle, AWS, Azure, or WatsonX offer collaborative access, facilitating community interaction, and resource sharing, including easy access to powerful computational resources like GPUs and TPUs. On the other hand, local environments/tools such as Anaconda are installed on a local machine, providing autonomy and control over data resources [72].

Figure 3.1: Platform environments that host Jupyter notebooks

Alternatives to Jupyter notebooks are Integrated Development Environments (IDEs) like Visual Studio Code (VSCode) or specialized environments like MATLAB. While VSCode offers a comprehensive set of features such as debugging, version control, and integrated terminals, making it suitable for structured and large-scale projects. MATLAB provides a distinctive environment tailored for numerical computing, featuring interactive interfaces, powerful visualization tools, and dedicated toolboxes for machine learning and computer vision tasks.

Ultimately, the environment choice depends on factors such as project scale, resource requirements, budget, security needs, and data sensitivity.

3.1.1 Python Programming Language

Python is a high-level, interpreted programming language designed for code readability through the use of significant indentation, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It supports procedural, structured, functional, and object-oriented programming paradigms while hosting a comprehensive standard library, as advertised in its website (Figure 3.2).

Originating in the late 1980s, Python first release in 1991 was Python 0.9.0. Python 2.0 launched in 2000, with the final release, Python 2.7.18, in 2020. Python 3.0, introduced in 2008, was a major revision not fully backward-compatible with prior versions. Currently the last stable release is Python 3.12, featuring active support [73].

Python has emerged as the dominant language in the data science and machine learning domains for several reasons. Its simplicity and readability make it accessible to both beginners and experienced developers, allowing for faster prototyping and experimentation. The language's fast learning curve has enabled a robust and active developer community that contributes significantly to various machine learning projects, resulting in extensive documentation, tutorials, and a wealth of readily available resources for practitioners. Python also offers an extensive array of libraries and frameworks specifically designed for computer vision, such as OpenCV, TensorFlow, PyTorch, and Keras, which simplify complex tasks and streamline the development process.

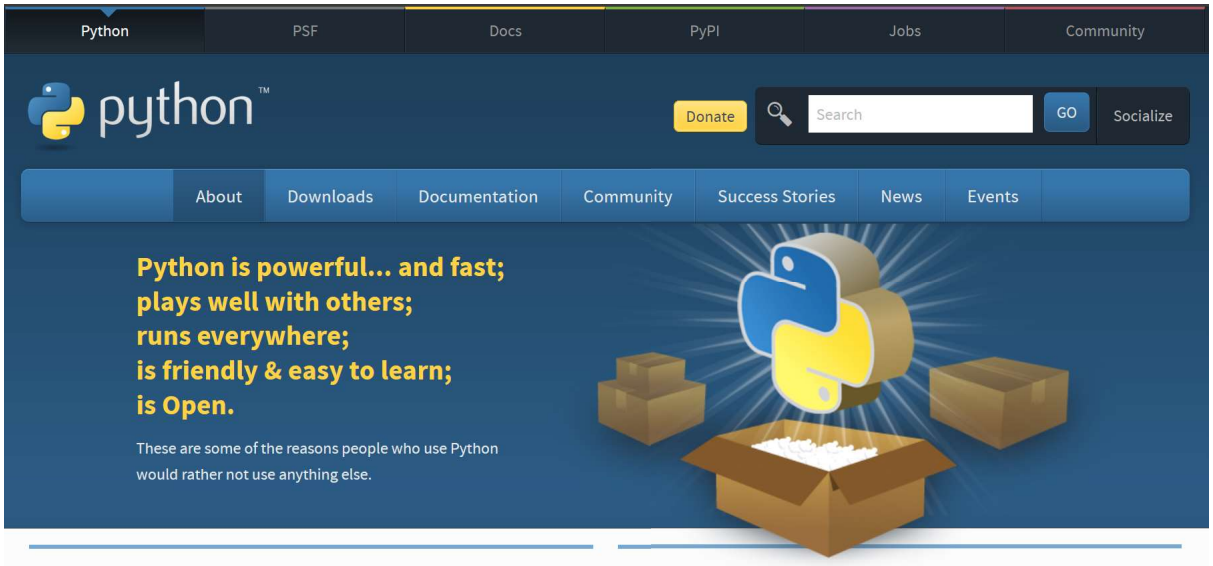


Figure 3.2: Python website homepage

Besides Python, other languages like C++, Java, or MATLAB, also find use in specific niches within computer vision and deep learning. For instance, compiled languages like C++ might be preferred for their performance in applications where speed is crucial, while MATLAB might be favored for its strong mathematical computing capabilities and specific toolboxes [74].

3.1.2 Anaconda Navigator

The elected platform for this project is Anaconda, chosen for its suitability with the scale of the project. Given the project's relatively small scope and absence of cross-collaboration needs, Anaconda aligns well with the project's priorities. Anaconda's autonomy, offline capabilities, and flexibility, which allow to work independently while ensuring privacy and control over the developing environment contributed for the choice. These aspects are crucial for the project's requirements, enabling to maintain a self-contained and versatile workspace.

Anaconda is a popular open-source distribution of the Python programming language. Equipped with its own package manager called Conda, which efficiently manages package dependencies and ensures compatibility with other packages and libraries installed within the same environment, helping prevent conflicts between packages and simplifies the management of data science libraries and tools.

Additionally, Anaconda provides the capability to create isolated environments for various projects, each with specific packages and versions tailored to the project's requirements. This isolation ensures that modifications or updates in one project do not impact others, and it allows for documenting of exact package versions used for future reproducibility. Figure 3.3 shows the Windows GUI that enables environment management.

Furthermore, the integration of other data science libraries and tools that are part of the Anaconda distribution, including NumPy, pandas, scikit-learn, and more, guarantees that Jupyter Notebook shares the same Python environment as these libraries, resulting in improved compatibility and performance.

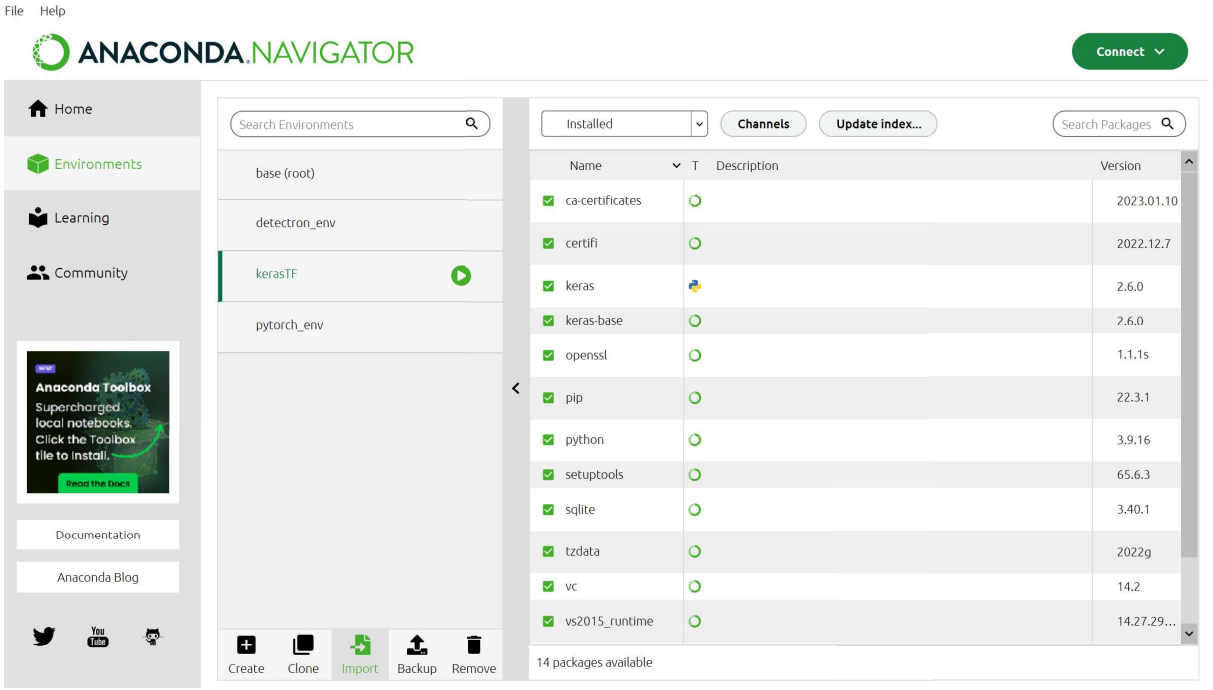


Figure 3.3: Anaconda Navigator environment management interface

3.1.3 Jupyter Notebook

Major cloud computing providers have embraced the use of Jupyter Notebook or its derivatives as the primary frontend interface for cloud users. The growth in Jupyter Notebook usage is remarkable. In 2015, approximately 200,000 Jupyter notebooks were available on GitHub, soaring to around 2.5 million by 2018. By January 2021, this number surged to nearly 10 million, showcasing a substantial increase in usage and contributions. As of July 2022, the Jupyter extension for VS Code has been downloaded over 40 million times, establishing itself as the second-most popular extension in the VS Code Marketplace [75].

The interactive and simple interface of Jupyter Notebooks has been widely adopted by:

- **Data Scientists and Analysts** for exploratory data analysis, visualizations, and insights sharing.
- **Researchers** for documenting and sharing reproducible research and analysis.
- **Developers** primarily for prototyping, experimenting, and documenting code and workflows in a collaborative environment.
- **Educators** to create interactive, easily shareable teaching materials.

The main benefits that Jupyter Notebooks offer are:

- **Interactivity and Visualization:** Allowing an interactive computing environment where code, visualizations, and explanations can be seamlessly integrated.
- **Ease of Sharing:** Notebooks can be easily shared, combining code, results, and descriptions in a single document.
- **Flexible Development:** They support different programming paradigms, enabling users to work procedurally, in a structured manner, or following object-oriented principles.

However, there are challenges that Jupyter notebooks do not address, like:

- **Version Control:** Tracking changes in notebooks through version control systems like Git can be challenging due to the JSON-based structure of notebooks, which can lead to merge conflicts.
- **Reproducibility:** Ensuring complete reproducibility of results can be tricky, especially when a notebook relies on external dependencies or stateful execution.
- **Performance:** While excellent for prototyping and exploration, Jupyter Notebooks might not be the best choice for computationally intensive tasks where performance optimization is crucial.

An example of the Jupyter Notebook interface running on a local machine through the Anaconda Navigator is presented in Figure 3.4.

jupyter Untitled (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
space = np.zeros([5,5], int)  # create a empty matrix of 5x5 dimension
print(space)
```

```
[[0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]
 [0 0 0 0 0]]
```

In [3]:
```
line1 = np.identity(5, int)   # create an identity matrix of 5x5 dimension
line2 = np.rot90(line1)       # create a rotated matrix
```

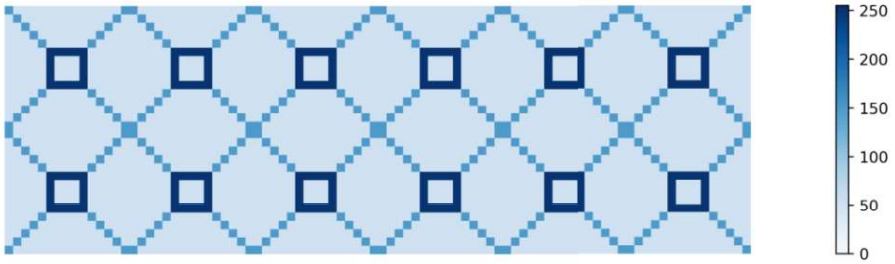
In [4]:
```
square = 2 * np.array([[1,1,1,1,1],     # elementwise multiplication by 2
                       [1,0,0,0,1],     # of a crafted matrix
                       [1,0,0,0,1],
                       [1,0,0,0,1],
                       [1,1,1,1,1]], int)
```

In [5]:
```
y1 = np.hstack((line1, space, line2))    # stack matrix horizontal
y2 = np.hstack((space, square, space))
y3 = np.hstack((line2, space, line1))
y4 = np.vstack((y1,y2,y3))               # stack matrix vertiacl
y5 = np.concatenate((y4, y4, y4, y4, y4), axis=1)
y6 = np.concatenate((y5, y5), axis=0)    # stacking alternative
```

In [6]:
```
img = y6 * 100 + 50     # elementwise multiply and ofset
img                     # similar to perform the print(img)
```

Out[6]:
```
array([[150,  50,  50, ...,  50,  50, 150],
       [ 50, 150,  50, ...,  50, 150,  50],
       [ 50,  50, 150, ..., 150,  50,  50],
       ...,
       [ 50,  50, 150, ..., 150,  50,  50],
       [ 50, 150,  50, ...,  50, 150,  50],
       [150,  50,  50, ...,  50,  50, 150]])
```

In [7]:
```
plt.figure(figsize = (20, 3), dpi=500)  # control the size and resolution
img_plot = plt.imshow(img, cmap='Blues', vmin=0, vmax=255)
plt.colorbar(img_plot)                  # prevent automatic normalizaion
plt.axis('off'); plt.plot;
```



In [ ]:

Figure 3.4: Demonstration of the Jupyter Notebook interface

3.1.4 Libraries and APIs

Depending on project requirements, time constraints, and the balance between customization and efficiency, building functions from scratch can be a viable solution, but often avoided because using libraries and APIs, such as the popular ones presented in Figure 3.5, offers several advantages:

- **Efficiency:** Libraries and APIs provide pre-built functionalities, saving time and effort in coding complex operations or algorithms.
- **Reliability:** Established libraries and APIs are often tested, refined, and maintained by a community.
- **Scalability:** Designed to be versatile and adaptable to various data types, structures, and use cases.
- **Abstraction:** By utilizing existing solutions, developers can focus on the core aspects of their project rather than reinventing the wheel, leading to faster development cycles.
- **Community Support:** Popular libraries and APIs have a strong user community, providing support, documentation, and potential collaboration opportunities.

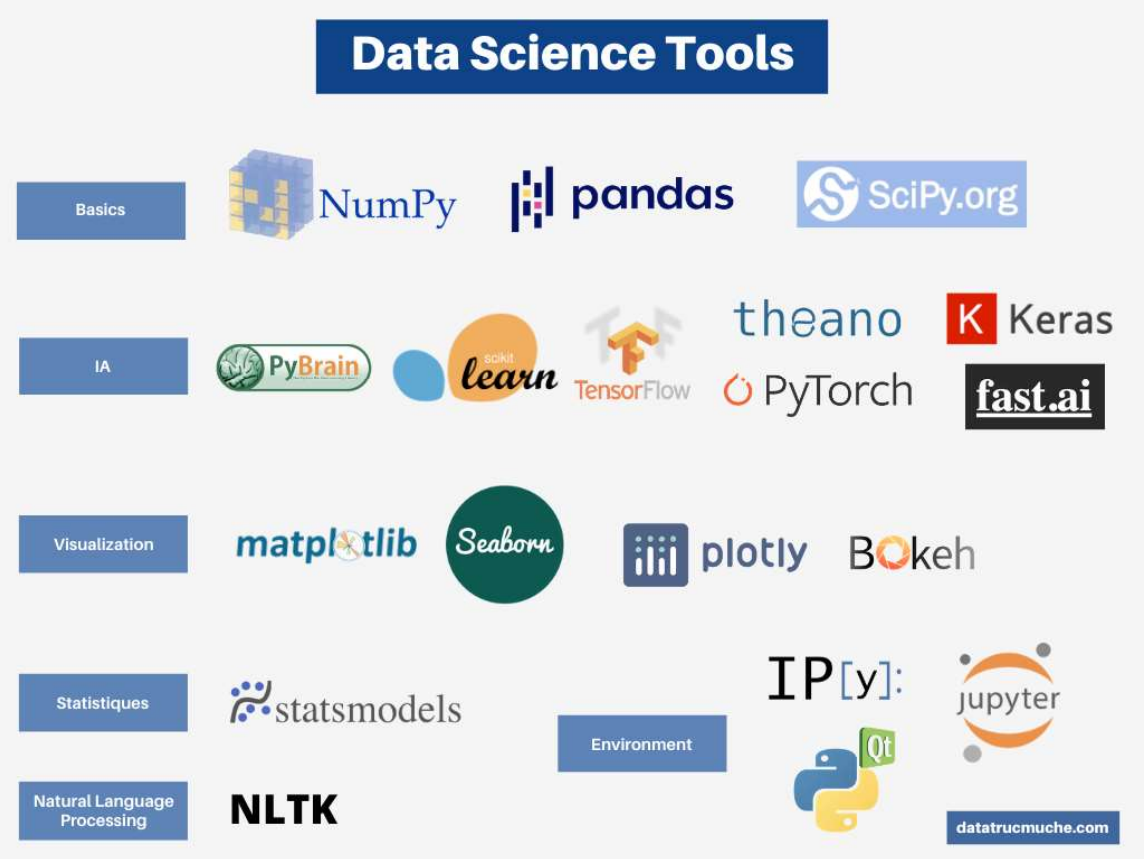


Figure 3.5: Popular ML libraries and APIs

In the following list, the main libraries utilized in this project for conducting data analysis, machine learning, and computer vision tasks, are shortly described:

- **NumPy** is a fundamental library for numerical computing in Python, supporting multidimensional arrays, matrices, and a collection of mathematical functions, enabling efficient operations on large arrays of data.
- **Pandas** is a powerful library for data manipulation and analysis through data structures like DataFrames and Series, simplifying tasks such as data cleaning, manipulation, and exploration.
- **Scikit-learn** includes a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and tools for model selection, evaluation, and preprocessing of data.
- **Keras** is a high-level neural networks API designed for easy and fast experimentation with deep learning models, providing a user-friendly interface and supporting various backends, including TensorFlow and Theano.
- **TensorFlow** offers a comprehensive ecosystem for building and deploying machine learning models, primarily focused on deep learning, providing tools for building neural networks and handling large datasets efficiently.
- **PyTorch** is a deep learning framework that allows for dynamic computation graphs, making it suitable for tasks such as natural language processing, computer vision, and reinforcement learning.
- **FastAI** is a high-level deep learning library built on top of PyTorch, offering simplified APIs and pre-built components for training state-of-the-art deep learning models while emphasizing ease of use and fast experimentation.
- **Pillow** provides extensive functionalities for image processing, editing, and manipulation, making it a widely used library for working with images in Python.
- **Matplotlib** is a widely used plotting library in Python, providing extensive support for creating static, interactive, and publication-quality visualizations, allowing users to create a wide range of plots and charts.
- **Seaborn** is another data visualization library in Python built on top of Matplotlib. It provides a higher-level interface for creating attractive statistical graphics, simplifying the creation of complex visualizations, especially for statistical analysis.
- **ImbLearn** is a library specifically designed for tackling imbalanced datasets commonly encountered in machine learning tasks. It provides various techniques for resampling, under-sampling, and over-sampling data to mitigate class imbalance issues, thereby improving model performance on imbalanced datasets.
- **Detectron2** is a deep learning framework built on PyTorch, specifically for object detection and segmentation tasks. It offers a modular design allowing users to mix and match various components such as backbones, feature extractors, and task-specific heads. It provides a comprehensive set of tools for training, evaluation, and inference, including utilities for data loading, visualization, and model management. It implements various state-of-the-art object detection algorithms known for their accuracy in detecting and segmenting objects in images.

3.2 Project Setup Showcase

In this section, the aim is to provide clear guidance through concise explanations and important details, that are complemented by practical examples supported with comprehensive references for further information, such as the O'Reilly hands-on ML [70].

To run the following scripts, it is essential to first import the libraries in Listing 3.1.

Listing 3.1: Libraries used along this project setup showcase

```
import os
import glob
import numpy as np
import seaborn as sn
import tensorflow as tf
import matplotlib.pyplot as plt

import visualkeras
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import Dense
from tensorflow.keras import regularizers
from tensorflow.keras import initializers
from tensorflow.keras import applications
from tensorflow.keras import callbacks
from tensorflow.keras import preprocessing
from tensorflow.keras.models import Model
from tensorflow.keras.utils import plot_model

from sklearn.preprocessing import label_binarize
from sklearn.metrics import roc_curve, auc, confusion_matrix
from imblearn.metrics import classification_report_imbalanced
```

3.2.1 Data Preparation

Dataset splitting is the initial step in model development, by preventing data leakage to ensure unbiased model evaluation. When dealing with smaller datasets, a common practice involves dividing it into 80% for training, 10% for testing, and another 10% for validation. However, as the dataset size scales up to hundreds of thousands or millions of images, the training portion often increases to 90% or even 95%.

Shuffling the dataset before splitting helps prevent any inherent order, ensuring that each subset accurately represents the dataset's characteristics. However, if we consistently perform random shuffling and splitting the entire dataset each time we run the code and reload the all data into RAM, we risk assessing a model using data previously employed for training, leading to biased performance evaluations.

A common practice is organizing the data splits in separated directories, ensuring clear segregation between subsets and the uniqueness of the testing dataset split. It is recommended to verify the existence of representative data across all subset splits.

Listing 3.2 presents an excerpt of a working code with the main arguments used.

Listing 3.2: Dataset importation using tensorflow [76]

```
dataset_path = 'datasets/'
my_seed = 12345
my_image_size = (224,224)
my_batch_size = 64

# LOAD THE TRAINING DATASET FROM A SEPARATED DIRECTORY
train_dataset = preprocessing.image_dataset_from_directory(
    directory = dataset_path + 'train',
    label_mode = 'categorical', # 'binary' # 'int' # 'None'
    seed = my_seed,
    image_size = my_image_size,
    batch_size = my_batch_size,
)

# LOAD A TESTING DATASET AND SPLIT 50% FOR VALIDATION
test_dataset, valid_dataset = preprocessing.image_dataset_from_directory(
    directory = dataset_path + 'test',
    label_mode = 'categorical', # 'binary' # 'int' # 'None'
    validation_split = 0.5,
    subset = "both",
    seed = my_seed,
    image_size = my_image_size,
    batch_size = my_batch_size,
)
```

A balanced dataset refers to an even distribution of samples across all classes, ensuring an equitable representation for each class. The code provided in Listing 3.3 facilitates the classes count assessment through a bar graph visualization.

Listing 3.3: Count and visualize the data distribution over classes

```
# VISUALIZE THE CLASSES COUNT ON TRAINING DATASET

class_names= train_dataset.class_names
count_by_class_train = [0] * len(class_names)

for images, labels in train_dataset:
    label_indices = tf.argmax(labels, axis= 1).numpy()
    for index in label_indices:
        count_by_class_train[index] + = 1

fig, ax = plt.subplots(figsize= (8, 4), dpi= 100)
bars = ax.barh(class_names, count_by_class_train)
ax.bar_label(bars);
```

Using an imbalanced dataset for classification tasks can result in suboptimal training outcomes. There is a risk that the model might predominantly predict the majority class for most instances, minimizing the overall loss but neglecting the minority classes overshadowing in the learning process, affecting the model's ability to discern significant patterns and generate precise predictions across all classes.

To illustrate that problem, an extreme example on a binary classification task can be considered, with a dataset of 100,000 images where only 1,000 images belong to the defective class. In such case, if a model predicts that all images have no defects, achieving 99% accuracy becomes meaningless, since the model is worthless.

The final dataset tip, is focused on the implementation of data augmentation.

To tackle the common issue of limited image availability, data augmentation can be employed on-the-fly, without generating new files, only by adding preprocessing layers that manipulate the images. This can be done by using a lambda function [77] to perform specific operations on the data, as exemplified in Listing 3.4.

Listing 3.4: Dataset augmentation using pre-processing layers [78]

```
# DATA AUGMENTATION USING PRE-PROCESSING LAYERS

data_augmentation = keras.Sequential(
    [
        layers.RandomFlip("horizontal_and_vertical", seed = my_seed),
        layers.RandomRotation(factor = 0.5, fill_mode = 'nearest', seed = my_seed),
        layers.RandomZoom(height_factor = (0, 0.1), width_factor = (0, 0.1),
                                    fill_mode = 'nearest', seed = my_seed),
        layers.RandomContrast(factor = 0.1, seed = my_seed),
        layers.RandomBrightness(factor = 0.1, seed = my_seed),
        layers.RandomCrop(height = 200, width = 200),
        layers.Resizing(224, 224) # input compatible with ImageNet trained weights
    ]
)

augmented_train_ds = train_dataset.map(lambda x, y:
                                        (data_augmentation(x, training= True), y))
```

This method ensures that the model encounters the same number of distinct samples, and consequently, the count of back-propagation steps during training does not directly increases. As a result, the method potentiates the advantages of diverse variations derived from augmenting the training data, without imposing additional demands on computational resources, such as time and power. This approach assists the model in learning a wider array of patterns and enhances its overall robustness.

3.2.2 Architecture Definition

One way to define a DL architecture is by sequentially arranging the layers, as demonstrated in the code shown in Listing 3.5.

Listing 3.5: Create a custom Neural Network using sequential layers [79] [80]

```
my_input_shape= (224,224,3)
num_classes= len(class_names)

# CREATE A CUSTOM MODEL USING SEQUENTIAL LAYERS

custom_model = keras.Sequential([
    layers.InputLayer(my_input_shape),
    layers.Rescaling(scale = 2.0/ 255.0, offset = -1.0), # Normalization Layer

    # Convulutional Block
    Conv2D(filters= 16, kernel_size= 7, strides= 2, padding= 'valid', activation= 'relu'),
    Conv2D(filters= 32, kernel_size= 5, strides= 2, padding= 'valid', activation= 'relu'),
    Conv2D(filters= 32, kernel_size= 5, strides= 2, padding= 'valid', activation= 'relu'),
    Conv2D(filters= 64, kernel_size= 3, strides= 2, padding= 'same', activation= 'relu'),
    Conv2D(filters= 64, kernel_size= 3, strides= 2, padding= 'same', activation= 'relu'),
    Conv2D(filters= 128, kernel_size= 3, strides= 2, padding= 'same', activation= 'relu'),
    layers.Flatten(),

    # Fully Connected Block
    Dense(256, activation= 'relu'),
    Dense(128, activation= 'relu'),
    Dense(64, activation= 'relu'),
    Dense(num_classes, activation= 'softmax') #'sigmoid'
])

custom_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
                     metrics = ['accuracy']) # 'binary_crossentropy'

custom_model.summary()
plot_model(custom_model, show_shapes= True)
```

Another method is to employ transfer learning, where a pre-trained model is imported, and its final classification layer is replaced with a customized layer tailored to the specific number of classes in the problem. Listing 3.6 demonstrates the implementation of this process.

Listing 3.6: Import and adapt a pre-trained model [81] [82]

```
# USE A LITERATURE MODEL PRE-TRAINED
base_model = tf.keras.applications.VGG16(weights= 'imagenet', include_top= False,
                                         input_shape= (224, 224, 3))
for layer in base_model.layers:
    layer.trainable = False # Freeze the pre-trained layers
```

```
x = base_model.output
x = Dense(128, activation= 'relu')(x) # Add a fully connected layer for customization
predictions = Dense(num_classes, activation= 'softmax')(x)

literature_model = Model(inputs= base_model.input, outputs= predictions)

literature_model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
                      metrics = ['accuracy']) # 'binary_crossentropy'
literature_model.summary()
visualkeras.layered_view(literature_model)
```

After preparing the dataset and defining the architecture, the training process can be initiated. This involves to learn the CNN parameters, the weights and biases, that minimize the difference between predictions and labels, also known as Loss Function.

After training, if overfitting is detected, one solution is to introduce penalties into the loss function. This involves integrating regularizers directly into specific layers and recompiling the model. Another method to address overfitting is by adding Dropout layers, which randomly discard some trainable parameters during each backpropagation step.

Additionally, symptoms of underfitting can be addressed by initializing the model's weights to aid convergence during training. However, the reduced complexity of the model might be the real cause. The code snippet in Listing 3.7 demonstrates how these strategies can be implemented.

Listing 3.7: Adding regularizers into model's layers [83]

```
k_reg= regularizers.L1L2(l1= 1e-5, l2= 2e-4)
b_reg= regularizers.L2(1e-4)
a_reg= regularizers.L2(1e-5)

k_ini= initializers.RandomNormal(seed= my_seed, mean= 0., stddev= 0.1)
b_ini= initializers.Zeros()

# ADD REGULARIZERS AND INITIALIZERS
some_model = keras.Sequential([
    # Assuming some initial layers

    Conv2D(filters= 64, kernel_size= 3, strides= 1, padding= 'same', activation= 'relu',
                kernel_regularizer= k_reg, bias_regularizer= b_reg,
                activity_regularizer= a_reg,
                kernel_initializer= k_ini, bias_initializer= b_ini),
    layers.Dropout(0.1),

    # Repeat for some other layers
    # Assuming some final layers
])
```

3.2.3 Performance Assessment

Before going into the actual training, it is wise to define some callbacks. The model checkpoint callback allow trained weights to be imported, either for future code runs or to restoring optimal parameters when encountering overfitting. Additionally, they aid in controlling the learning process by allowing early stopping if there is no improvement or by reducing the learning rate as the training process progresses towards the final stages. Examples of these techniques can be found in Listing 3.8.

Listing 3.8: Model Callbacks [84]

```
weights_path= 'callbacks/model_1/weights'
os.makedirs(weights_path, exist_ok= True) # Create the directory if it doesn't exist

# CREATE MODEL CHECK POINT AND OTHERS CALLBACKS
my_callbacks = [
    callbacks.ModelCheckpoint(filepath= os.path.join(weights_path,
                                                     'model{epoch:02d}_loss{val_loss:.2f}.h5'),
                                                     save_best_only= True,
                                                     save_weights_only = True),

    callbacks.EarlyStopping(monitor= 'val_loss', patience= 4,
                                                     verbose= 1,
                                                     min_delta= 0.001,
                                                     mode= 'min'),

    callbacks.ReduceLROnPlateau(monitor= 'val_loss', factor= 0.5,
                                                         verbose= 1,
                                                         cooldown= 1,
                                                         min_delta= 0.0005,
                                                         patience= 2,
                                                         min_lr= 0.00005)
]
```

To train the defined model using the imported dataset, it is as simple as calling the "fit" method of the class "model", as demonstrated in Listing 3.9.

Listing 3.9: Training the model to fit the dataset [85]

```
initial_epoch = 0
epochs = 25

# FIT THE MODEL TO THE TRAINING DATAST
history = custom_model.fit( train_dataset, # or augmented_train_ds,
                            validation_data= valid_dataset,
                            epochs= epochs,
                            initial_epoch= initial_epoch,
                            callbacks= [my_callbacks]
)
```

To track the training process trend, such as improvement or divergence, it is very useful to visualize the history data returned by the previous function.

This can be accomplished by executing the code provided in Listing 3.10.

Listing 3.10: Ploting the performance history of the model during training [86]

```
# PLOT THE TRAINING PERFORMANCE

fig, (ax1, ax2) = plt.subplots(1, 2, figsize= (15, 5), dpi= 150)

ax1.plot(history.history['loss'], label= 'Training Loss')
ax1.plot(history.history['val_loss'], label= 'Validation Loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')
ax1.set_title('Training and Validation Loss')
ax1.legend()

ax2.plot(history.history['accuracy'], label= 'Training Accuracy')
ax2.plot(history.history['val_accuracy'], label= 'Validation Accuracy')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')
ax2.set_title('Training and Validation Accuracy')
ax2.legend()

plt.tight_layout()
plt.show()
```

When it is found that the results are potentially satisfactory, this iterative process is moved forward by loading the saved weights and evaluating the accuracy across all dataset splits by running the code in Listing 3.11.

Listing 3.11: Loading the saved parameters and evaluate the accuracy [87]

```
# LAOD THE WEIGHTS BEFORE OVERFITING

best_epoch = 5

file_name = f"model-{best_epoch:02d}_loss????.h5"
file_dir = glob.glob(os.path.join(weights_path, file_name))

best_model= custom_model
best_model.load_weights(file_dir[0])


# EVALUATE THE MODEL ACCURACY

best_model.evaluate(train_dataset)
best_model.evaluate(valid_dataset)
best_model.evaluate(test_dataset)
```

It might be useful to record that the purpose of a testing dataset is to evaluate the model without using its feedback to redefine it. This function is reserved for the validation dataset, and is the reason for its existence.

The main used metrics can be looked at by executing the code in Listing 3.12.

Listing 3.12: Reporting the main used metrics [88]

```
# Extrat the data from keras format to numpy array format
x, y = zip(* train_dataset); X_train = np.concatenate(x); y_train = np.concatenate(y)
x, y = zip(* valid_dataset); X_val = np.concatenate(x); y_val = np.concatenate(y)
x, y = zip(* test_dataset); X_test = np.concatenate(x); y_test = np.concatenate(y)

# Save the predictions probabilities
y_train_pred = best_model.predict(X_train)
y_val_pred = best_model.predict(X_val)
y_test_pred = best_model.predict(X_test)

print(classification_report_imbalanced(np.argmax(y_train, axis= 1),
                                       np.argmax(y_train_pred, axis= 1)))
```

The $argmax$ method used in the $numpy$ arrays, in Listing 3.12 and Listing 3.14, is only required for multi-class tasks that use sparse matrices.

To investigate more detailed assessments, the ROC curve can be obtained and analyzed through the execution of the code in Listing 3.13.

Listing 3.13: Obtaining the ROC curve in multiclassification One-vs-Rest [89] [90]

```
# Define the positive classes
positive_class = 4

# Binarize the labels to set a class as positive (1) and the rest as negative (0)
y_binary = label_binarize(y_train, classes= np.arange(len(class_names)))
y_binary = y_binary[:, positive_class]

# Calculate ROC curve
fpr, tpr, thresholds = roc_curve(y_binary, y_train_pred[:, positive_class])
roc_auc = auc(fpr, tpr) # Calculate the area under the ROC curve (AUC)

# Plot the ROC curve
plt.figure(figsize= (6, 5), dpi= 100)
plt.plot(fpr, tpr, lw= 2, label= 'ROC curve (area = {:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], lw= 2, linestyle= '--')
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)')
plt.legend(loc= 'lower right')
plt.show()
```

The confusion matrix can be obtained by using the code provided in Listing 3.14.

Listing 3.14: Obtaining and displaying the confusion matrix[91] [92]

```
# Create a Confusion Matrix
cm = confusion_matrix(np.argmax(y_val, axis= 1), np.argmax(y_val_pred, axis= 1))

# Display the Confusion Matrix
plt.figure(figsize = (10,7), dpi= 100)
ax = sn.heatmap(cm, annot= True, fmt= ".0f",
                            cmap = "PuRd",
                            xticklabels= class_names,
                            yticklabels= class_names)

ax.set(xlabel= "PREDICTIONS", ylabel= "LABELS");
```

To explore the model performance even further, tools like Grad-CAM [93], SHAP [94] and LIME [95] can be used, as well as other explainer algorithms [96].

The code provided in Listing 3.15 is the implementation of Grad-CAM, assuming the definition of the function *make_gradcam_heatmap* [97].

Listing 3.15: Implementation of Grad-CAM explainer [97]

```
# List all Convolutional layer of the model
all_layers = [layer.name for layer in best_model.layers]
conv2d_layers = [layer for layer in all_layers if layer.startswith('conv2d')]

for i in range(30): # Loop over 30 images
    image = X_val[i] # Get a single image from X_val
    image = tf.expand_dims(image, 0) # Expand dimensions to create a batch axis

    plt.figure(figsize= (15, 5), dpi= 100)
    for j, name in enumerate(conv2d_layers): # Loop over all Conv2D layers
        heatmap = make_gradcam_heatmap(image, best_model, name)

        plt.subplot(1, len(conv2d_layers), j + 1)
        plt.imshow(heatmap, cmap= 'hot')
        plt.title(f'Layer: {name}')
        plt.axis('off')

    # Get the model's prediction probabilities for this image
    prediction_probs = best_model.predict(image)
    predicted_class_index = tf.argmax(prediction_probs, axis= 1)[0].numpy()
    true_class = class_names[np.argmax(y_val, axis= 1)[i]]
    predicted_class = class_names[predicted_class_index]

    plt.suptitle(f"True Class: {true_class}, Predicted Class: {predicted_class}")
    plt.subplots_adjust(top= 1.3)
    plt.show()
```

4 Case Studies and Implementations

This chapter presents implementations of Automated Visual Inspection of Assets (AVIA), with real-world data, using the previously described framework and tools. These results from practical applications are crucial for validating the potential of a deep learning approach for AVIA and uncover the source of the current limitations.

The first case study is categorized as a binary classification task, where the operational condition of the asset is classified as defective or functional. The used assets are Photovoltaic (PV) Cells, and the data used to describe its condition are Electro-Luminescence (EL) images. It is accepted that the information on those EL images holds a strong correlation with its operational condition. The methodology employed is known as transfer learning, as a pre-trained model was used to expedite the tuning process.

The second case study is focused on a multi-class classification task, where multiple operational failures are diagnosed based on the thermal patterns of the assets. The assets, represented by Infra-Red (IR) thermographic images, are PV Modules. Due to the data characteristics, such as low resolution and reduced/simple features, several shallow custom CNNs trained from scratch were employed to evaluate the impact of network architecture hyperparameter selection.

The last case study addressed an object detection task, where different assets are localized and classified in each image. The considered assets are High-Voltage (HV) Insulators, more precisely, each individual ceramic disc. The condition of each asset is assessed by the RGB image representation of the operational environment they are in. State-of-the-art deep learning models were employed and compared to evaluate the performance trade-offs involved.

Utilized images, representing the assets considered, are illustrated in Figure 4.1.

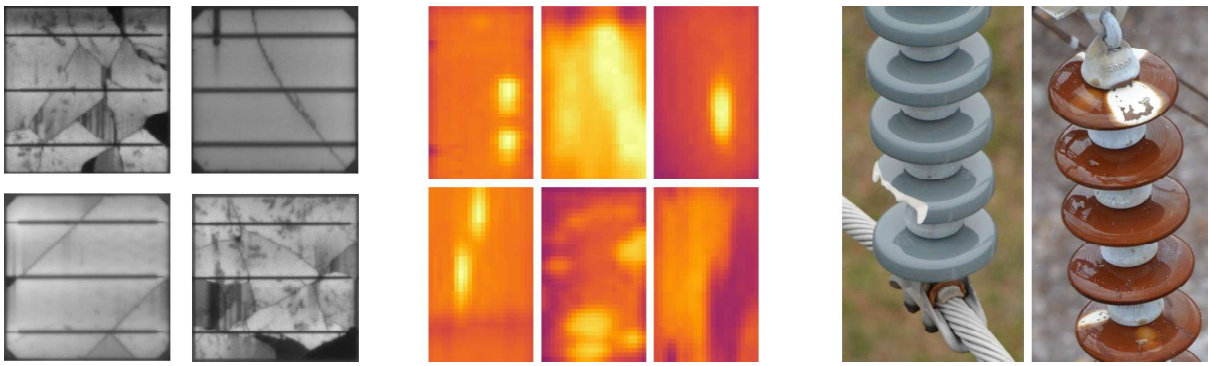


Figure 4.1: Example images of PV Cells [98], PV Modules [99], and HV Insulators [100]

4.1 Binary Classification of Photo-Voltaic Cells

Monitoring and maintaining the integrity of solar modules is crucial for ensuring optimal performance in photovoltaic systems. The visual identification of defects poses a challenge, even for trained experts. Not all conditions affecting a module's efficiency are visible, and conversely, visible flaws don't always impact efficiency. To address this, non-destructive methods such as IR and EL imaging offer promising solutions.

IR imaging detects damaged cells by capturing the infrared radiation emitted when solar cells stop converting energy into electricity due to disconnection from the circuit. However, the resolution limitations of IR cameras might hinder the detection of minute defects, such as microcracks that haven't yet impacted conversion efficiency. On the other hand, EL imaging offers higher resolution, revealing premature defects. While effective, EL imaging requires trained specialists and manual inspection, making it time-consuming and costly. However, the potential for automated defect classification in EL images presents an opportunity to overcome these challenges.

The principle of electroluminescence involves applying a voltage bias to a solar cell, generating an electric field within the cell. Defects, cracks, or non-uniformities in the silicon material can lead to localized variations in the electric field. When carriers (electrons and holes) recombine in these areas, they emit light, a phenomenon known as electroluminescence. Dark regions or patterns in electroluminescence images may indicate defects, cracks, or other irregularities. EL imaging is employed during the manufacturing process to identify defects early on, ensuring the production of high-quality solar cells. In the field, EL imaging can be used to troubleshoot and identify the causes of reduced performance or failures in installed solar panels.

Figure 4.2 illustrates the five main Cell defects that can be found in EL images: (a) Shows a solar cell exhibiting a typical material defect. (b) Illustrates interruptions in the indicated areas, known as finger interruptions, which may not necessarily reduce module efficiency. (c) Displays a solar cell containing a subtle microcrack. Although these microcracks do not completely divide the cell, detecting them is crucial as they have the potential to expand over time. (d) Demonstrates a disconnected area resulting from cell interconnection degradation. (e) Presents a cell with electrically separated or degraded segments, typically caused by mechanical damage [101].

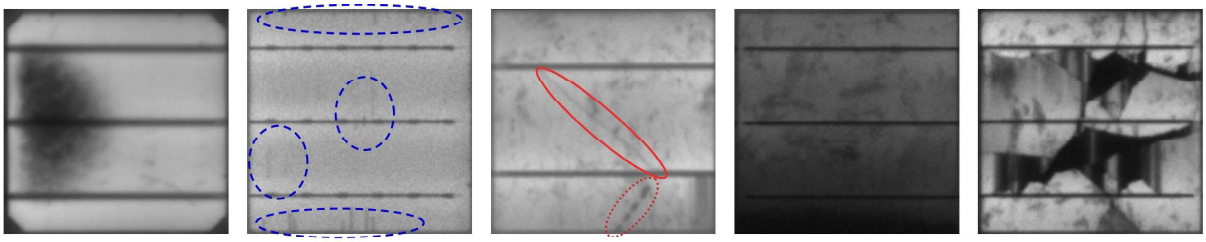


Figure 4.2: Different intrinsic and extrinsic defects in solar cells [101]

4.1.1 Related Work

Sergiu Deitsch *et al.* introduced a fully automatic segmentation method [102] designed to precisely extract solar cells from high-resolution EL images. This segmentation method proves robust against underexposure and performs effectively even in the presence of severe defects on solar cells. This robustness can be attributed to the proposed preprocessing techniques and the integration of ridgeness filtering, alongside tensor voting, to accurately determine inter-cell borders and busbars. Notably, the segmentation method exhibits high accuracy (97.80%), enabling its output to be used for subsequent inspection tasks, including automatic classification of defective solar cells and the prediction of potential power loss.

In their other study [101], they compared two processing variants. Both approaches are trained using 75% of the 2,624 solar cell images extracted from high-resolution EL intensity images of monocristallyne and polycrystalline PV modules. The hardware-efficient approach uses hand-crafted features classified in a Support Vector Machine (SVM), while the more hardware-demanding method employs an end-to-end deep Convolutional Neural Network (CNN) running on Graphics Processing Units (GPU). The pipeline for the SVM classifier was crafted through experiments involving various Dense Sampling and Keypoint Detection methods, revealing that the optimal configuration involves utilizing KAZE/VGG features with a linear Support Vector Machine (SVM). The CNN was a fine-tuned regression network derived from VGG-19. It demonstrates higher accuracy, averaging 88.42%, with an inference time of 12 minutes on a CPU. In comparison, the SVM achieves a slightly lower average accuracy of 82.44% but completes inference on the testing dataset in 8 minutes using the same hardware.

Haenara Shin *et al.*[103] propose machine learning-based automated classification strategies for detecting defects in PV cells using EL images. Initially, they employed conventional classification methods, such as SVM and Random Forest (RF), achieving up to 64% accuracy. Subsequently, they explored standard CNN models, using transfer learning, with experiments revealing a 92% accuracy on ResNet50 as a binary-class classifier.

Lawrence Pratt *et al.*[104] introduced a benchmark dataset with corresponding ground truth masks for multi-class semantic segmentation in EL images of solar PV cells and the performance metrics from four semantic segmentation models trained using three sets of class weights. The model's performance was assessed using metrics like median Intersection over Union (mIoU) and median recall (mRcl) for a subset of common defects (cracks, inactive areas, and gridline defects) and features (ribbon interconnects and cell spacing) present in the dataset. While the mIoU for cracks is low (25%) the recall is high (86%), and the resulting prediction masks reliably locate the defects in complex images with both large and small objects, proving useful in the context of detecting cracks and other defects in EL images.

4.1.2 Data-Set

The dataset [98] contains 2,624 samples of 300x300 pixels 8-bit grayscale images of functional and defective solar cells with varying degree of degradation, extracted from 44 different solar modules, more precisely, 26 monocrystalline and 18 polycrystalline.

Given the unavailability of actual measurements for power degradation, a pseudo target variable was generated based on the assessment made by electroluminescence experts and considering their confidence levels in each cell image.

A label of 1.0 was assigned when the expert was highly confident that the cell image contained a defect. For evaluations of defects with lower confidence, a label of 0.667 was utilized. Functional cells assessed with high confidence were labeled as 0.0, while those assessed with less confidence received a label of 0.333.

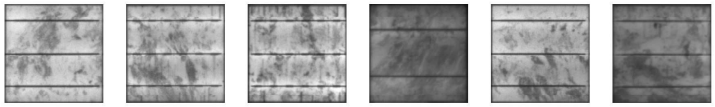
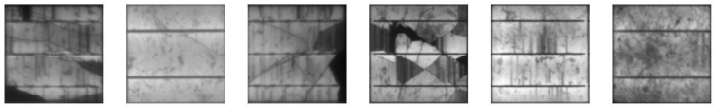
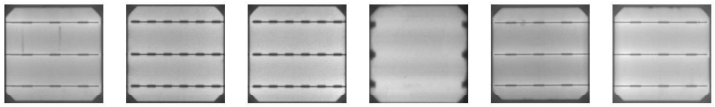
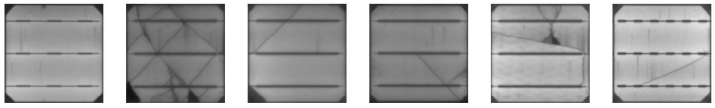
In regression tasks, the target variable is typically related to a continuous domain. In this specific context, although numeric, these values represent varying confidence levels, and can be interpreted as discrete categories for different confidence levels rather than a strictly continuous variable.

To approach this problem as a binary classification task, assessments with lower confidence, representing 15% of the original dataset, were excluded. The resulting dataset comprises only two classes, categorizing images as either functional or defective.

Additionally, the dataset was divided by cell type: one dataset for monocrystalline and another for polycrystalline solar cells.

Table 4.1 presents the quantities and sample images from the considered datasets.

Table 4.1: Quantity and visual description of the PV Cells dataset classes

Type	**Class**	**Images Quantity**	**Image Samples**
Poly	Functional	920	
Poly	Defective	402	
Mono	Functional	588	
Mono	Defective	313	

Around 20% of the data was reserved in a separate directory for validation and testing purposes, resulting in the quantities presented bellow. The information extracted from Table 4.2 also reveals a class imbalance of approximately 70% to 30%.

Table 4.2: Quantity of images on the binary dataset splits

Type	Class	Training Split		Validation & Testing Split	
		Quantity	Class Ratio	Quantity	Class Ratio
Poly	Functional	690	69,6%	230	69,7%
Poly	Defective	302	30,4%	100	30,3%
Mono	Functional	442	65,3%	146	65,2%
Mono	Defective	235	34,7%	78	34,8%

Although this value is typically considered indicative of an imbalanced dataset, it is not necessarily critical for model fitting and was not a main focus in this initial experience.

However, this issue can be addressed in various ways. Some advanced methods include replacing Binary Cross-Entropy (BCE) with other functions like the Focal Loss [105] or the Contrastive Loss [106]. A simpler way to address data imbalance is by employing oversampling and/or undersampling. The safest approach involves producing augmented images for the under-represented classes and/or discarding images from the over-represented class. Other known methods, such as Synthetic Minority Oversampling Technique (SMOTE) [107], are used to generate data based on identified patterns. However, they are considered a risky choice in this case, since they have the potential to alter our data specificities.

4.1.3 Methodology and Architecture

Acknowledging that the dataset has a relatively small quantity of images, data augmentation was applied in the pre-processing layers. This means it is performed only during model fitting and not during model inference. To prevent the potential erasure of crucial features, zooming or cropping augmentation it was avoided. Additionally, to ensure the preservation of EL image properties, adjustments in contrast and brightness were also avoided. This implies that only rotations and mirrorings have been applied.

The selected architecture was VGG19 [56] because of its relative simplicity and easy to understand, while the 19-layer depth of the network enables it to capture hierarchical features. Additionally, it was originally trained on the ImageNet dataset, which includes a large number of images spanning a wide variety of categories. This choice led to better overall performance and saved considerable training time and computational resources.

Employing VGG19 also allows for a direct comparison of results with the work presented in the literature by Sergiu Deitsch *et al.* [101], which have used the same data.

Figure 4.3 presents The VGG19 model architecture, using the *Visualkeras* tool [82].

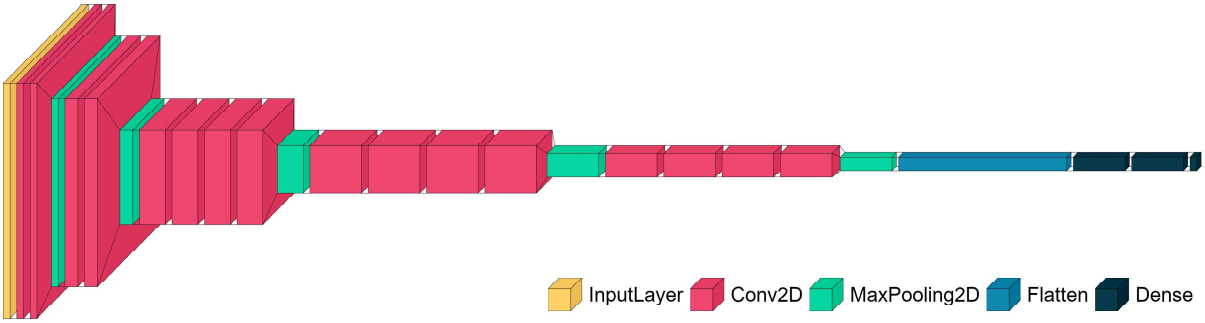


Figure 4.3: VGG19 architecture used for PV Cell classification

The implementation involved using only the original feature detector, also known as *Convolutional Block*, along with Imagenet's respective weights. While the original classifier was replaced with a custom *Dense Block*, composed with a *Flatten Layer*, two *Dense Layers* of 512 neurons each and *ReLU Activation*, and ending in a single-node *Dense Layer* using *Sigmoid Activation*. The weights for these dense layers were randomly initialized with a mean of 0,0 and a standard deviation of 0,1. The distribution of trainable parameters across the layers is summarized in Table 4.3.

Table 4.3: Summary of VGG19 models' architecture and parameters distribution

Block	Layer	Type	# Layer	Output Shape	# Parameters
0	0	InputLayer	-	(224, 224, 3)	0
1	1-2	Conv2D	2	(224, 224, 64)	38 720
	-	MaxPooling2D	1	(112, 112, 64)	0
2	3-4	Conv2D	2	(112, 112, 128)	221 440
	-	MaxPooling2D	1	(56, 56, 128)	0
3	5-8	Conv2D	4	(56, 56, 256)	2 065 408
	-	MaxPooling2D	1	(28, 28, 256)	0
4	9-12	Conv2D	4	(28, 28, 512)	8 259 584
	-	MaxPooling2D	1	(14, 14, 512)	0
5	13-16	Conv2D	4	(14, 14, 512)	9 439 232
	-	MaxPooling2D	1	(7, 7, 512)	0
6	-	Flatten	1	(25088,)	0
	17	Dense	1	(512,)	12845568
	18	Dense	1	(512,)	262656
	19	Dense	1	(1,)	513

Regularization in the form of L1 and L2 loss penalty was applied to the dense layers' coefficients, with a starting value of 1×10^{-5} and 1×10^{-5}, respectively.

Additionally, three dropout layers were inserted before each dense layer, with dropout rates of 40%, 30%, and 20%, respectively.

The training process consisted of several stages, controlled by early stopping after 10 iterations with no validation loss progress. The next training stage began after loading the weights from the last best iteration. Starting with a high learning rate of 0.05 allowed the fitting process to explore weight values over a wider domain range. The learning rate was reduced by a factor of 5 for the next two stages, while the regularization was doubled. The fine-tuning stages included progressively unfreezing the final convolutional layers, while decreasing the learning rate to as low as 0.00001.

4.1.4 Results

The metrics presented from Figure 4.4 to Figure 4.6 summarize the results obtained by inferring the testing dataset split. In addition, Table 4.4 presents a comparison with the metrics obtained by Sergiu Deitsch et al. [101]. The True class is the functional one.

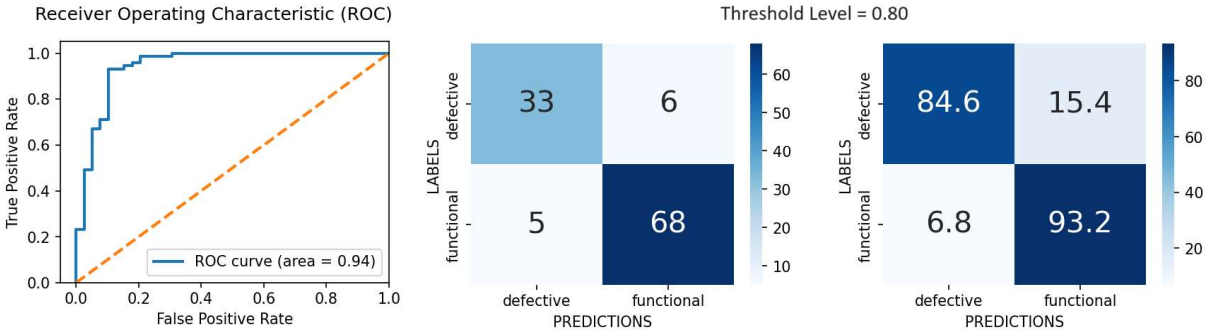


Figure 4.4: Monocrystalline PV Cells test inference ROC and Confusion Matrix

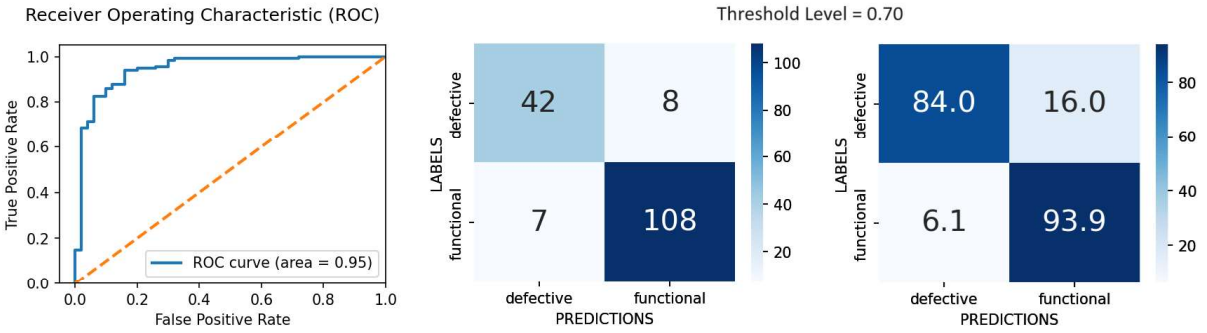


Figure 4.5: Polycrystalline PV Cells test inference ROC and Confusion Matrix

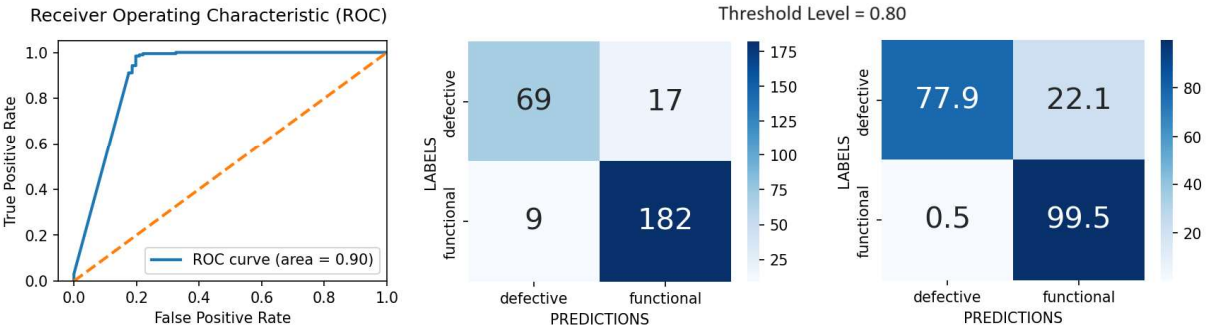


Figure 4.6: PV Cells Mix test inference ROC and Confusion Matrix

Table 4.4: PV Cells test inference results comparison to reference work [101]

Results	AUC			TP rate			TN rate		
	mono	**poly**	**mix**	**mono**	**poly**	**mix**	**mono**	**poly**	**mix**
Reference	**95,2%**	93,8%	**94,4%**	90,3%	86,1%	87,9%	82,8%	**86,0%**	**84,7%**
Presented	94%	**95%**	90%	**93,2%**	**93,9%**	**99,5%**	**84,6%**	84,0%	77,9%

4.1.5 Evaluation

Figure 4.7 and Figure 4.8 display the results obtained using Grad-CAM++ [108] [109], enabling a visual assessment of the input features that are most significant for the classification verdict. Additional examples are available in Appendix A.

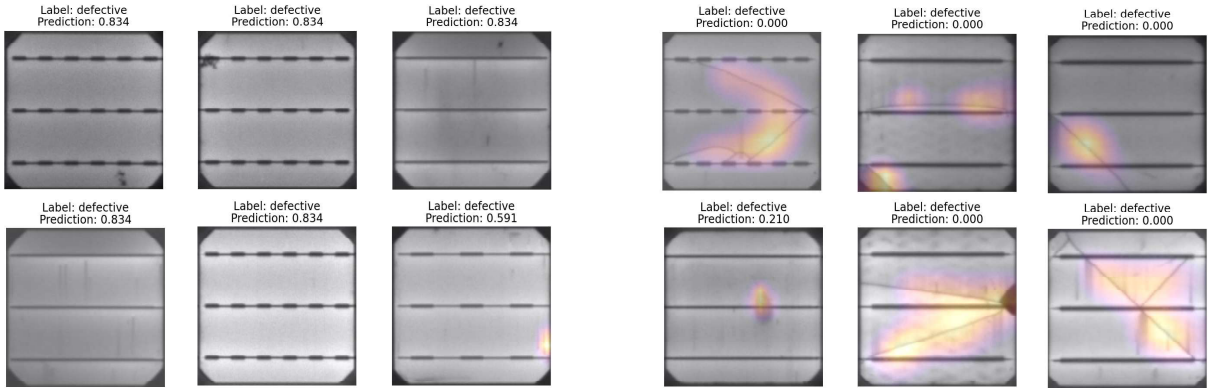


Figure 4.7: Defective Monocrystalline PV Cells test inference results and heatmap

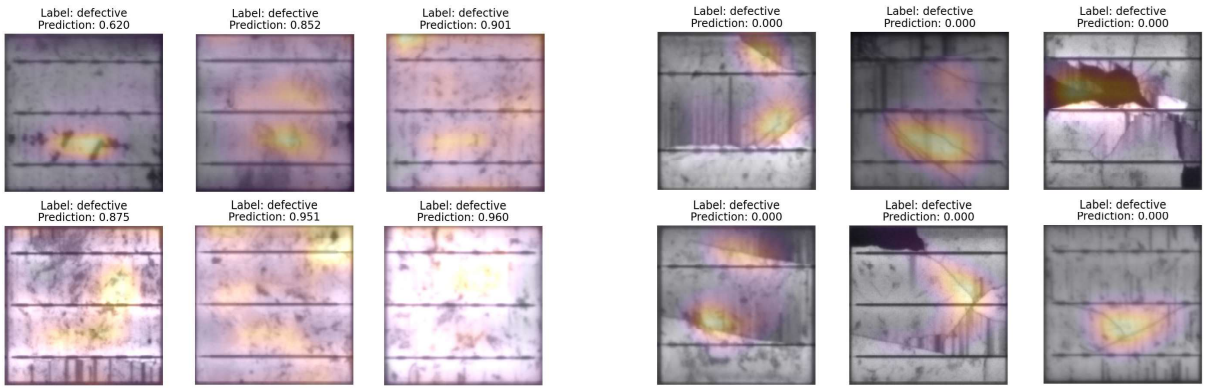


Figure 4.8: Defective Polycrystalline PV Cells test inference results and heatmap

It is worth noting that the misclassified images reveal small or imperceptible features. Finger interruptions were described as not causing efficiency loss, with numerous examples existing in the functional class. The presence of these images in the defective class could raise questions about the equality of criteria in the experts' assessments.

To be fair, the degradation level of a PV cell is not binary. There are different types of defects, each with its intrinsic visual features, and associated degradation effects.

4.2 Multi Classification of Photo-Voltaic Modules

A complete solution for inspecting a PV plant starts with data acquisition, including UAV route planning, followed by video frame cropping to create thermographic images of PV modules. These images are then categorized into different defect classes, ultimately leading to the elaboration of a maintenance report. In this study we focus on the central part of this pipeline system: the computer vision core functions as a sensor that interprets physical quantities and provides valuable classification information. Our system involves using cropped thermal images of PV modules as input data and generate predictions for each defect class, outputting the top predicted class. 4 custom CNN models were tested on an image dataset with samples presented in Figure 4.9.

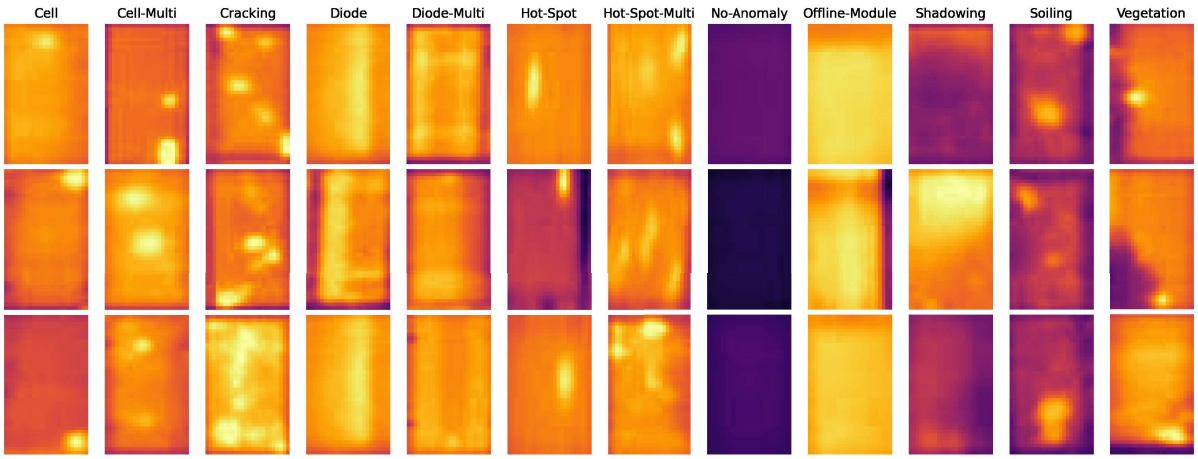


Figure 4.9: Sample of IR images in 12 classes

4.2.1 Related Work

In the study led by Yahya Zefri *et al.*[110], PV modules were segmented using images from UAVs. They categorized the modules into six groups: Non-defective module, One hotspot, Patchwork pattern of hotspots, Overheated module row, Overheated module, and Pointed heating. They then compared two models: one they fine-tuned from a pre-trained VGG16 model and another they built from scratch with five convolutional layers. Surprisingly, their own simple model outperformed the more complex VGG16 model, suggesting that the model complexity should match the data complexity.

Jacek Starzyński *et al.*[24] highlighted the impact of radiance deprivation effects, often caused shadows, on a PV module's power output. And used recent DL architectures, including YOLOv4, on a dataset of thermal images collected with UAVs, but revealing modest accuracy. This emphasized the need for data augmentation. The proposed strategy focused on enhancing the Recall score during training to reduce false negatives and perform real-time detections during PV farm overflights, with an adaptive flight path for additional shots from different angles, heights and directions, suggesting a method to address the common scarcity of data examples in defective classes.

4.2.2 Data-Set

In response to the limited availability of infrared imagery, an initiative by Raptor Maps, Inc. 2020 © introduced a labeled dataset of 20.000 images [99]. This dataset covers 12 classes of solar module anomalies, including 11 anomaly types and a 'No-Anomaly' class. Table 4.5 lists each class along with a description.

Table 4.5: Quantity and description of the dataset 12 classes [99]

Class	Images	Description
Cell	1,877	Hot spot occurring with square geometry in single cell.
Cell-Multi	1,288	Hot spots occurring with square geometry in multiple cells.
Cracking	941	Module anomaly caused by cracking on module surface.
Hot-Spot	251	Hot spot on a thin film module.
Hot-Spot-Multi	247	Multiple hot spots on a thin film module.
Shadowing	1,056	Sunlight obstructed by vegetation, man-made structures, or adjacent rows.
Diode	1,499	Activated bypass diode, typically 1/3 of module.
Diode-Multi	175	Multiple activated bypass diodes, typically affecting 2/3 of module.
Vegetation	1,639	Panels blocked by vegetation.
Soiling	205	Dirt, dust, or other debris on surface of module.
Offline-Module	828	Entire module is heated.
No-Anomaly	10,000	Nominal solar module.

Collected via piloted aircraft and unmanned aerial systems, the images vary in resolution from 3.0 to 15.0 cm/pixel. Each anomaly class was cropped and labeled, aided by corresponding visible spectrum images for classification accuracy.

Figure 4.10 illustrates a canonical example of each anomaly.

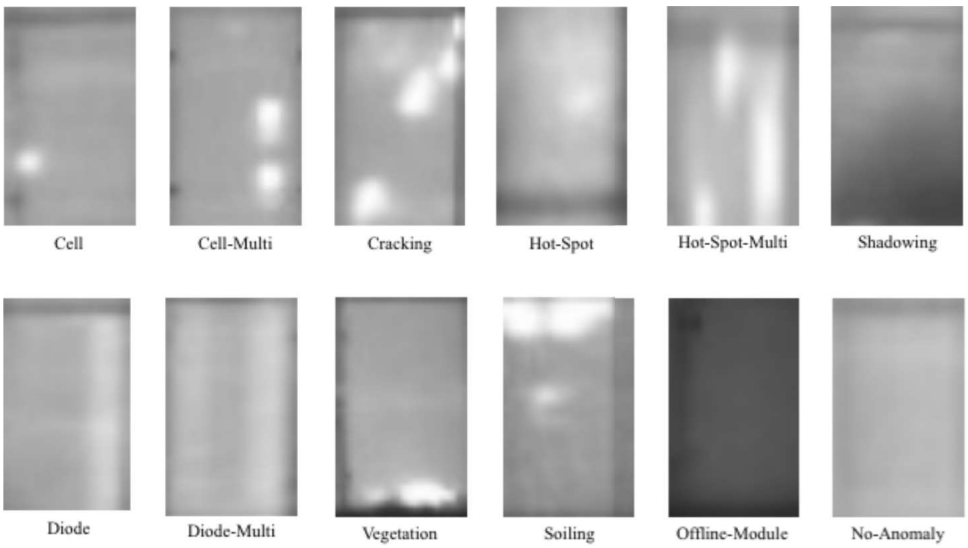


Figure 4.10: Examples of Canonical IR Images for the 12 Classes

We used a subset of 6000 images (80% training, 10% validation, 10% testing) for assessing the presented solution while we performed data augmentation on the under-represented classes of the original dataset.

From the original dataset, we randomly selected 150 images from each class, reserved 100 for validation and testing, and augmented the other 50 for training. The augmentation procedure involved only basic operations, such as rotation and mirroring, because other conventional transformations such as cropping and shearing, have the potential to modify crucial features.

This option resulted from the recognition that an imbalanced dataset, when used for multi-class classification, can lead to sub-optimal training outcomes. There is a risk that the model might predominantly predict the majority class for most instances, minimizing the overall loss but neglecting the minority classes, overshadowed in the learning process, and affecting the model's ability to discern significant patterns and generate precise predictions across all classes.

4.2.3 Methodology and Architecture

Ensuring feature consistency, we maintained the original 40x24 spatial resolution and the 8-bit amplitude resolution of the images in the dataset.

Our investigation involved a model architecture based on the work in [110], with structural adjustments to suit the used dataset. This architecture, referred to as M1 in Figure 4.15 and in the benchmarking Table 4.6, serves as a reference for performance comparisons.

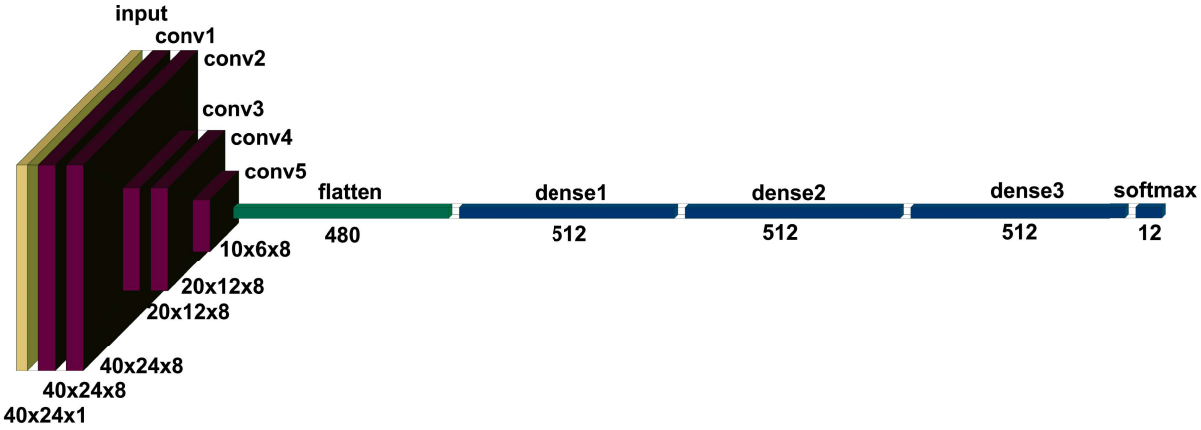


Figure 4.11: Tested model M1 architecture diagram

The tested models share a common structure: a block of convolutional layers followed by a block of dense layers, with a flatten layer in between. The first layer is a normalization layer, while the final dense layer utilizes *softmax* activation for classifying into the 12 predefined classes.

The convolutional layers use a kernel size of 3x3, with padding set to 'same,' and a stride of 1 to maintain output size or a stride of 2 to replace max-pooling layers. ReLU activation is applied to both Conv2D and dense layers. Figure 4.11 to Figure 4.14 present a comparison between the different tested architectures from model M1 to model M4.

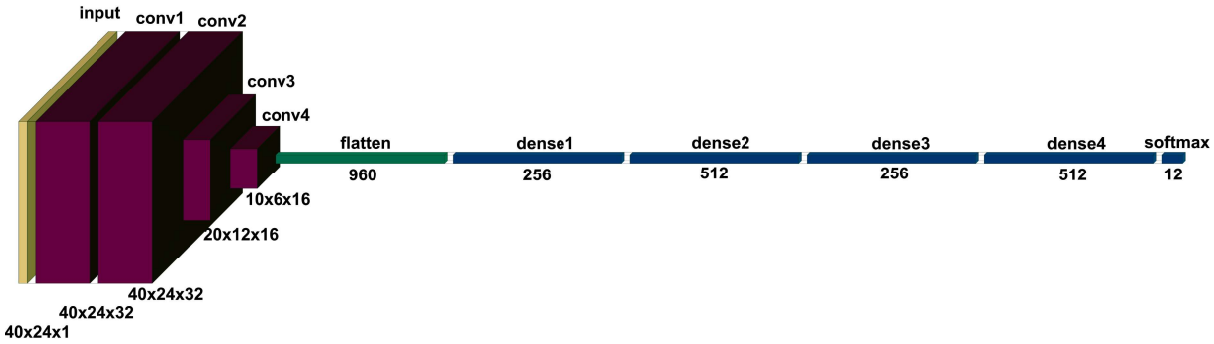


Figure 4.12: Tested model M2 architecture diagram

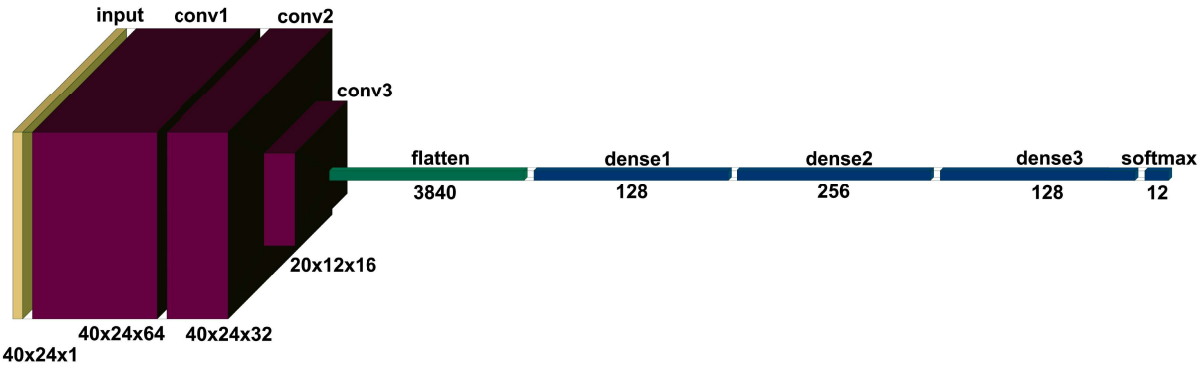


Figure 4.13: Tested model M3 architecture diagram

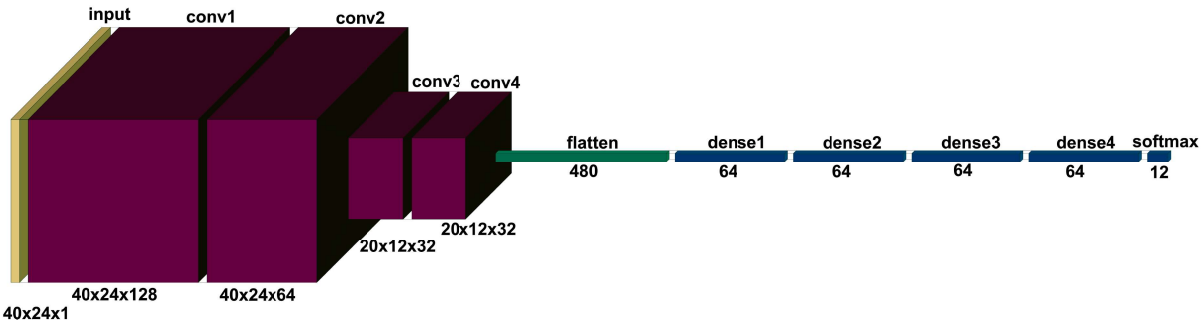


Figure 4.14: Tested model M4 architecture diagram

The leading direction was to increase the number of parameters in the convolutional block and decrease them in the dense block to hopefully achieve an optimal balance between the two.

4.2.4 Results

The evolution of the loss function as a result of the training procedure is presented in Figure 4.15. It can be observed that model M1 starts to overfit around the 20th epoch, while model M2 doesn't appear to overfit before the 50th epoch, but it also not converging. Model M3 begins to overfit after the 27th epoch but achieves the lowest loss value. M4 overfits around the 34th epoch, however does not outperform M3.

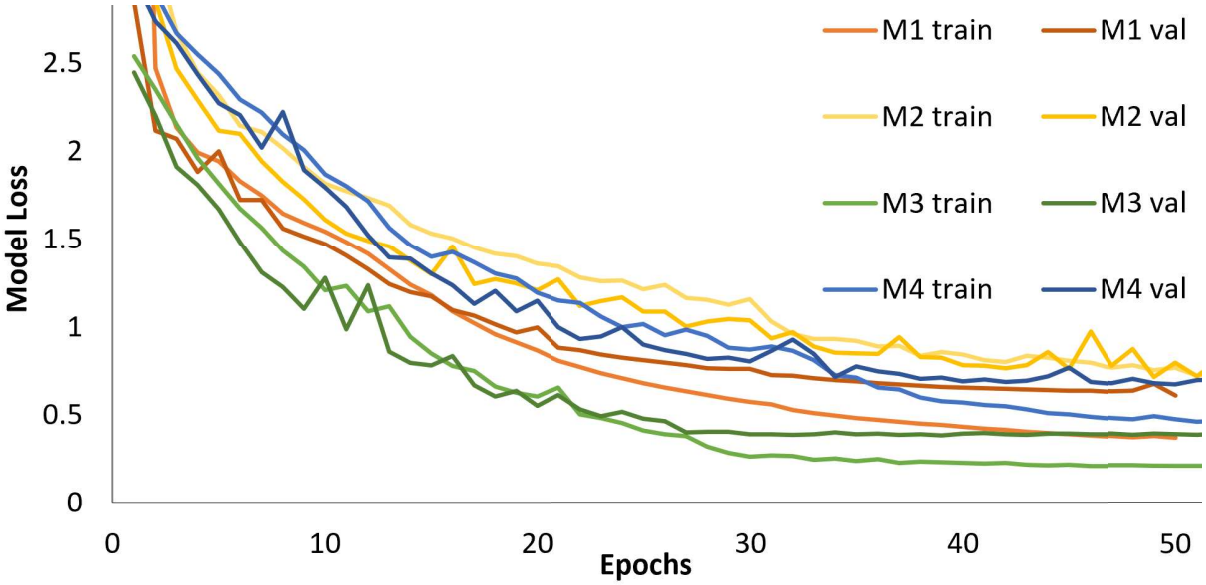


Figure 4.15: Evolution of the loss function over epochs

The performance of the tested models, subsequent to loading the corresponding weights before overfitting takes place, is summarized in Table 4.6 through the calculated Precision, Recall and F1-score.

Table 4.6: Number of defects identified by foot patrol and by the virtual inspection [18]

Model	Precision			Recall			F1-score		
	train	valid	test	train	valid	test	train	valid	test
M1	0.70	0.66	0.66	0.69	0.65	0.66	0.68	0.63	0.64
M2	0.74	0.68	0.70	0.72	0.67	0.70	0.72	0.66	0.69
M3	**0.92**	**0.88**	**0.87**	**0.93**	**0.88**	**0.86**	**0.92**	**0.88**	**0.86**
M4	0.84	0.75	0.79	0.83	0.74	0.79	0.83	0.74	0.78

In this application the Recall is considered more relevant since disregarding a false negative has a greater impact than responding to a false positive.

4.2.5 Evaluation

The obtained confusion matrix provides a comprehensive overview of the predictions made by model M3 on the test dataset split compared to the actual ground truth labels, as it can be seen in Figure 4.16.

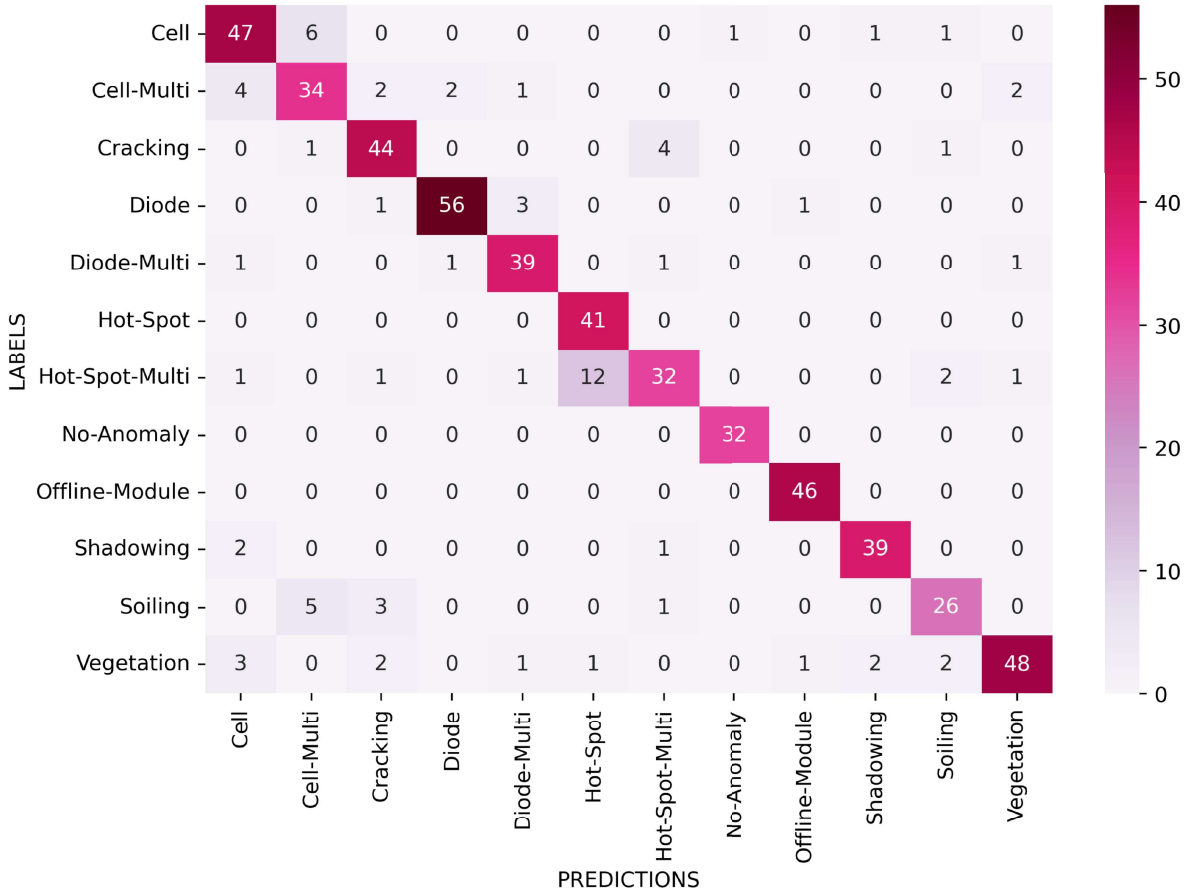


Figure 4.16: Confusion matrix of the best model (M3)

To obtain a comprehensive view of mispredicted images, across the classes with poor performance, we plotted randomly selected examples, comparing True Negatives (TN), False Negatives (FN), False Positives (FP) and True Positives (TP), across classes in Figure 4.17.

Upon examination, it is evident that the models struggle with accurate predictions when dealing with indistinct features or similar geometric patterns.

Further comparisons are available in Appendix A.

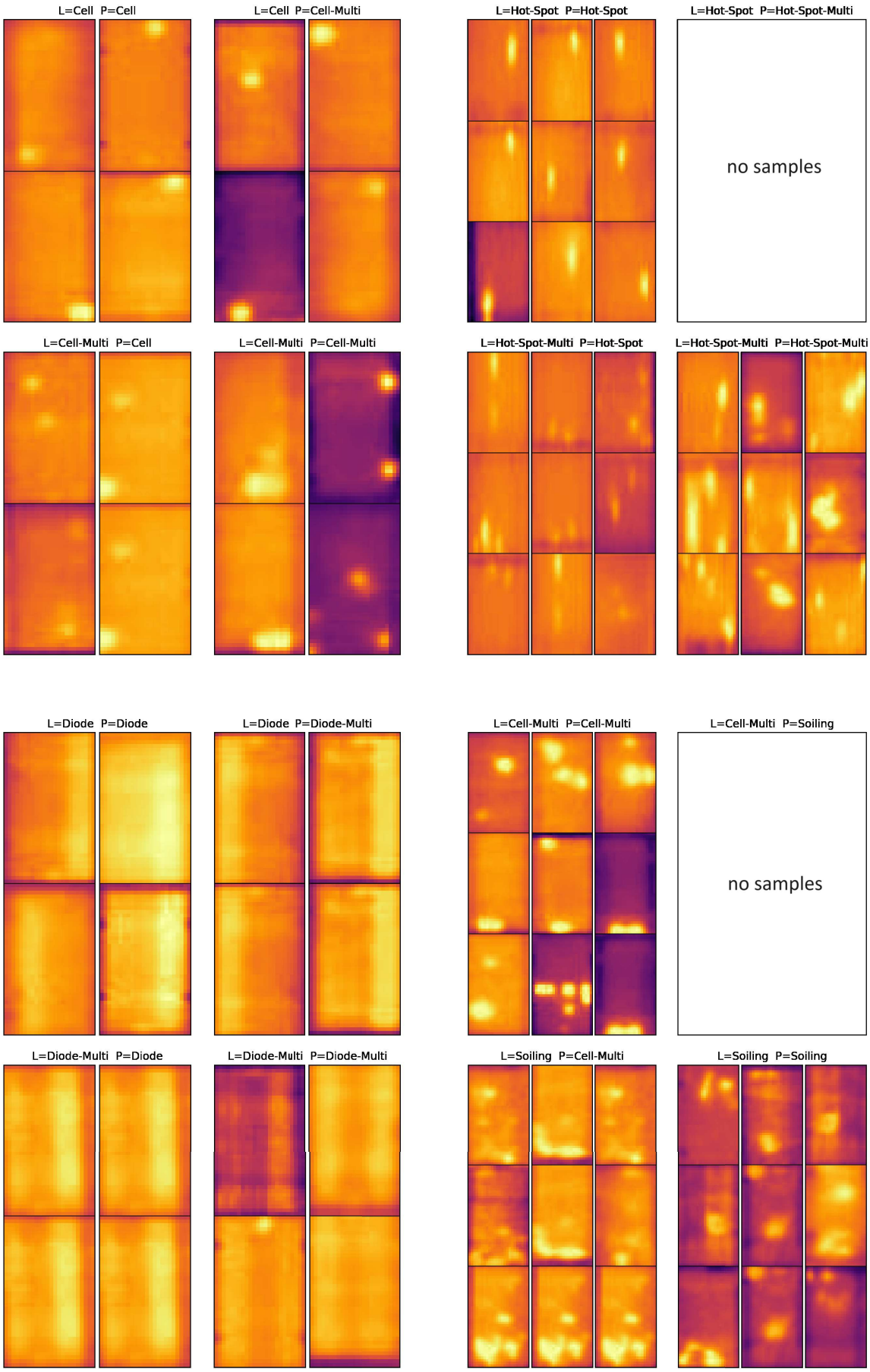


Figure 4.17: Image comparison of Label against Prediction over different classes

4.3 Object Detection of High-Voltage Insulators

The increasing volume of collected data such as images, during infrastructure inspections, and its processing, is currently being limited by human interpretation tasks, mainly because escalating this pipe-line element is expensive and prolonged. However, most inspection tasks with minor responsibilities can be automated using computer vision and deep learning techniques to support the system's growing demand.

This work focus on the core of an automated inspection pipeline, receiving high resolution RGB images, collected by UAV's in real life scenarios of electric power transmission and distribution, containing different types and amounts of electric insulators, and generates a localization bounding box with a respective class of the predicted defect.

The assets addressed here are the ceramic discs of each *Cap and Pin* insulator. Besides the default background class, the implemented model classifies each object as functional (good), mechanical defective disc (broken), or electrical defective shell (flashover). Figure 4.18 presents examples of images used to train the model.

Figure 4.18: Diversity of insulator images used for model training [100]

4.3.1 Related Work

Haiyan Cheng *et al.* [111] proposed a method for self-shattering defect detection of glass insulators based on spatial features using classical computer vision methods. The approach begins by applying a double-limit threshold in the RGB color space to achieve insulator segmentation and simple morphological operations to eliminate noise. The class evaluation involves analyzing the number of pixels in each Region of Interest (ROI), along with region length and the distances between regions, considering specified tolerance levels. Despite the fact that this and similar classical approaches achieve an accuracy exceeding 90%, these techniques do not demonstrate generalization across various insulator materials with differing colors.

Utilizing a dataset comprising over 2500 photographs collected from both studio and real-world outdoor environments, Ricardo M. Prates *et al.* [112] proposed a CNN classification model capable of automatically recognizing insulator conformity, learning from indoor photographs by augmenting them with realistic details like top ties and real-world backgrounds. Multi-Task Learning (MTL) was applied to enhance defect detection performance by simultaneously predicting the insulator class. The methodology achieves promising results, with a 92% accuracy in material classification, an 85% accuracy in defect detection with 75% F1-score.

To address uncertainty in manually labeled insulator datasets, Zhiyong Dai [113] introduces a novel object detection deep learning-based methodology, for bounding box regression and uncertainty estimation, entitled YOLOD which integrates a Gaussian prior into the YOLOX model. By predicting bounding box uncertainties and using them as weighting factors in the Non-Maximum Suppression (NMS) process, YOLOD improves the robustness of insulator defect predictions. Additionally, an improved KL loss optimizes the model's ability to represent the mean and variance of bounding box regression based on the Gaussian prior. Comprehensive experiments demonstrate that YOLOD outperforms benchmark models on insulator datasets, achieving a higher average precision of 73.9%, validating its effectiveness.

High-resolution aerial images captured by UAVs with intricate backgrounds and small target detection present a critical challenge for insulator defect detection. Qiaodi Wen *et al.* [8] overcome these complexities with two deep learning methodologies, Exact R-CNN and CME-CNN, both based on Faster R-CNN. Exact R-CNN incorporates advanced techniques such as Feature Pyramid Networks (FPN), Cascade Regression, and Generalized Intersection over Union (GIoU), along with innovative approaches like RoI Align and depthwise separable convolution to enhance accuracy while reducing computational demands. Additionally, CME-CNN introduces an insulator mask extraction network to eliminate background interference and employs Exact R-CNN for defect detection. Experimental results demonstrate the superior effectiveness of these methods, with CME-CNN-ResNet50 achieving an average precision of 88.7%.

4.3.2 Data-Set

The Electric Power Research Institute (EPRI) [19] has released an organized Insulator Defect Image Dataset (IDID), which is also published on *IEEE DataPort* [100].

This dataset comprises high-resolution annotated images of both healthy and defective porcelain insulators from high-voltage transmission lines. The images showcase a diverse range of colors, captured from various angles and lighting conditions.

A closer inspection reveals information that is not clear in the dataset instructions PDF file. As it can be seen in Figure 4.19, the 1600 annotated images files are, in fact, only 400 unique images along with 3 mirrored copies.

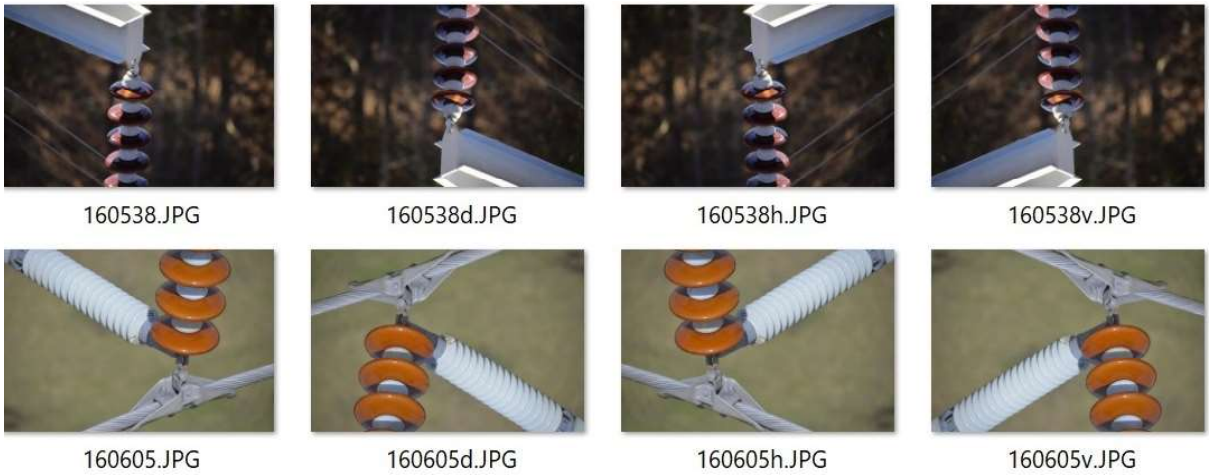


Figure 4.19: Mirrored copies in the annotated portion of the IDID [100]

This information led to avoid performing a random validation dataset split, as has been done by others [114]. Creating a validation directory with carefully selected annotated images ensured that no mirrored copies of the training dataset were included.

Non-annotated data was used for test inference with supervised evaluation, including the other 88 images available in the IDID, and a sample of 24 images of Medium Voltage (MV) power lines, from a vast private dataset collected by EDP LabElec [115].

The annotations are composed, essentially, by each object bounding box (x, y, w, h) and class label (Good, Broken, Flashover). The actual structure of the files depends on the standardized format chosen. In this work, the annotations were converted from the originally distributed format to MS COCO (Microsoft Common Objects in COntext) and YOLO (You Only Look Once) formats.

In the COCO format, all annotation data is encapsulated within brackets in a single JSON file, including image ID, category ID, and bounding box coordinates. YOLO uses separate directories for images and labels. The labels are stored in text files with the same name as their corresponding image. Each line of the text file contains 5 fields separated by a space character, representing the class ID and normalized coordinates.

Exactly 15% of the 1600 images, containing 17116 annotated instances, were used for validation, corresponding to 2488 objects over 240 images. Figure 4.20 illustrates the distribution of the annotated instances in the training and validation dataset splits.

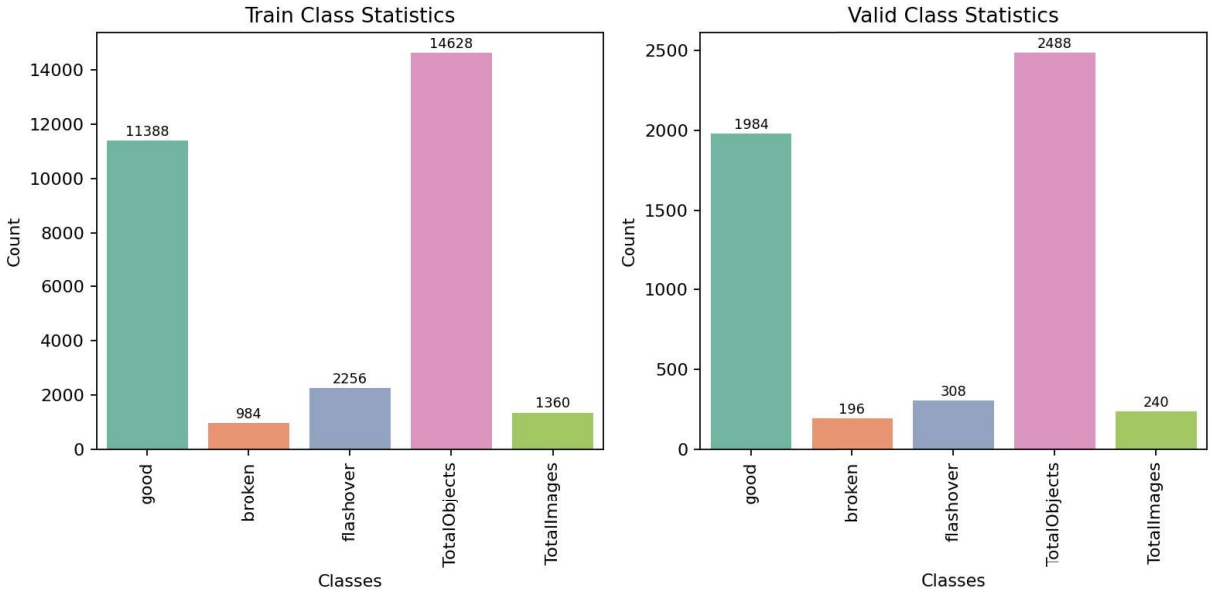


Figure 4.20: Counts of annotated objects in the training and validation dataset splits

All images in the dataset have high resolution, with a significant number ranging from 5 MP up to 15 MP. Table 4.7 details the properties of the dataset images. For a broad overview, the same information is shown in the histograms presented in Figure 4.21.

Table 4.7: Image properties overview of the IDID

Resolution	Aspect Ratio	Megapixels	Count
(1618, 1080)	1.50	1.75	4
(2144, 1424)	1.51	3.05	8
(2896, 1944)	1.49	5.63	20
(3008, 2000)	1.50	6.02	28
(3216, 2136)	1.51	6.87	364
(3680, 2456)	1.50	9.04	164
(3872, 2592)	1.49	10.04	88
(4288, 2848)	1.51	12.21	120
(4512, 3008)	1.50	13.57	432
(4928, 3264)	1.51	16.08	4
(3391, 5081)	0.67	17.23	4
(5520, 3680)	1.50	20.31	80
(6000, 4000)	1.50	24.00	260
(7360, 4912)	1.50	36.15	24

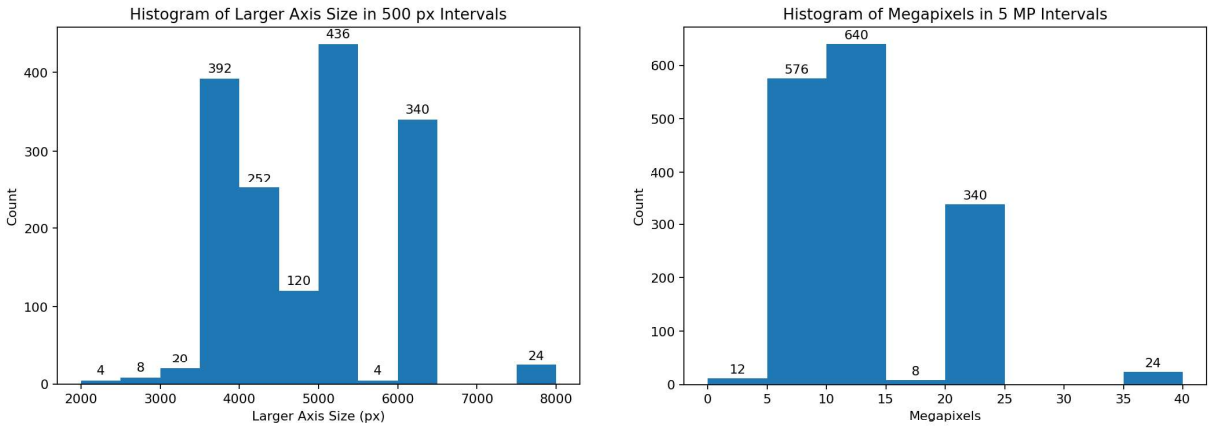


Figure 4.21: Image resolution histogram distribution of the IDID

Most object detection models resize input images during preprocessing layers, utilizing bilinear or bicubic interpolation, to ensure compatibility with the model's input size and architecture, which is often configurable.

Padding may be added to the resized images to maintain the original aspect ratio, thereby preventing distortion of objects in the image, during resizing.

After processing by the model, bounding boxes predicted for detected objects are resized accordingly to match the original size of the input image. This ensures that the bounding boxes accurately represent the locations of objects in the original image.

4.3.3 Methodology and Architecture

One of the goals of this project is to compare State-Of-The-Art (SOTA) models, particularly, the YOLOv8 one-stage detector and the Faster R-CNN two-stage detector.

For this purpose, the Ultralytics and the Detectron2 high-level frameworks were used, for employing the YOLOv8 and the Faster R-CNN models, respectively.

These high-level frameworks can be seen as repositories that provides implementations and utilities for common deep vision tasks, including object detection, instance segmentation, pose estimation, and others, as illustrated in Figure 4.22.

Both frameworks are built on top of PyTorch deep learning library.

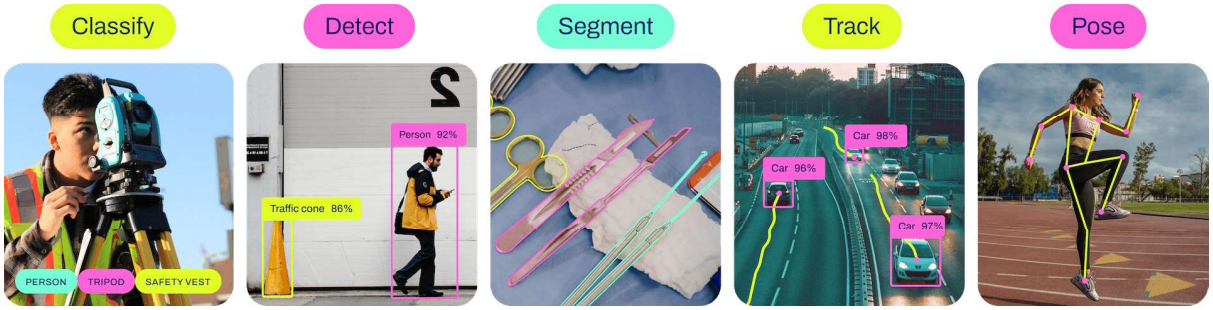


Figure 4.22: Common deep vision tasks addressed by high-level frameworks [64]

The variants of YOLOv8 for object detection range from the *nano* to the *extra-large* model, with their corresponding parameters and performance metrics presented in Table 4.8. YOLOv8 was also compared with prior versions in Figure 4.23.

Table 4.8: YOLOv8 model variants performance comparison [64]

Model	Size (pixels)	mAPval 50-95	Speed (ms) CPU ONNX	Speed (ms) GPU A100	Params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	**640**	**44.9**	**128.4**	**1.20**	**11.2**	**28.6**
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

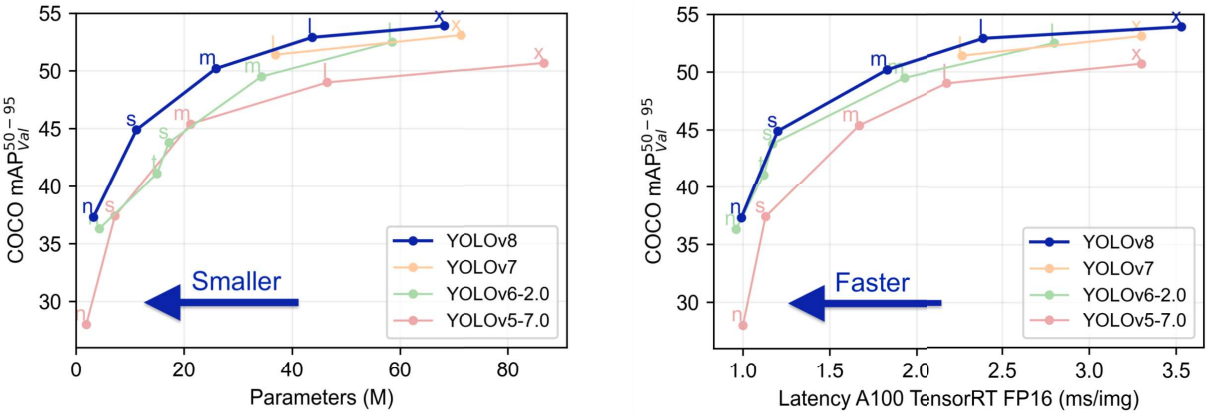


Figure 4.23: Accuracy-Speed trade-off on YOLO family [64]

Due to limitations regarding computational resources, the 16 GB Dual Tesla T4 GPU available on kaggle, it was employed the *small* version of YOLOv8, named *YOLOv8s*, which will be referred to as YOLO in the following section, for increased readability.

To better capture small defective visual features in the image data, the input size of the model was adjusted from the default configuration of 640 px to 1024 px, and was not further increased due to computational constrains.

The default configuration for data augmentation [116] was used, which randomly performs a variety of transformations, including adjustments to HSV (Hue, Saturation, Value), image translation, scaling, horizontal flipping, among others.

The batch size configuration was set to 32 images per batch. The learning rate was scheduled to start at 0.01 and decrease gradually by a factor of 1000 by the end of the training, which was set to 100 epochs.

In addition to the Ultralytics official documentation [64], one of the references for successfully conducting the YOLO implementation on a custom dataset was the work of Rafael Levy published on kaggle [117].

YOLOv8 has been built from the ground up incorporating all of the feedback and learnings from the highly successful YOLOv5, introducing several modifications to the original architecture to potentially enhance performance or address specific requirements.

It replaces the C3 module with the C2f module for feature extraction, swaps the initial 6x6 convolution in the backbone with a 3x3 convolution, removes two convolutional layers from designated positions in the network, substitutes a 1x1 convolution with a 3x3 convolution in the bottleneck layer, and adopts a decoupled head architecture while eliminating the objectness branch (Figure 4.24).

These changes aim to optimize feature extraction, streamline the network, and refine the prediction process for tasks like object detection [118].

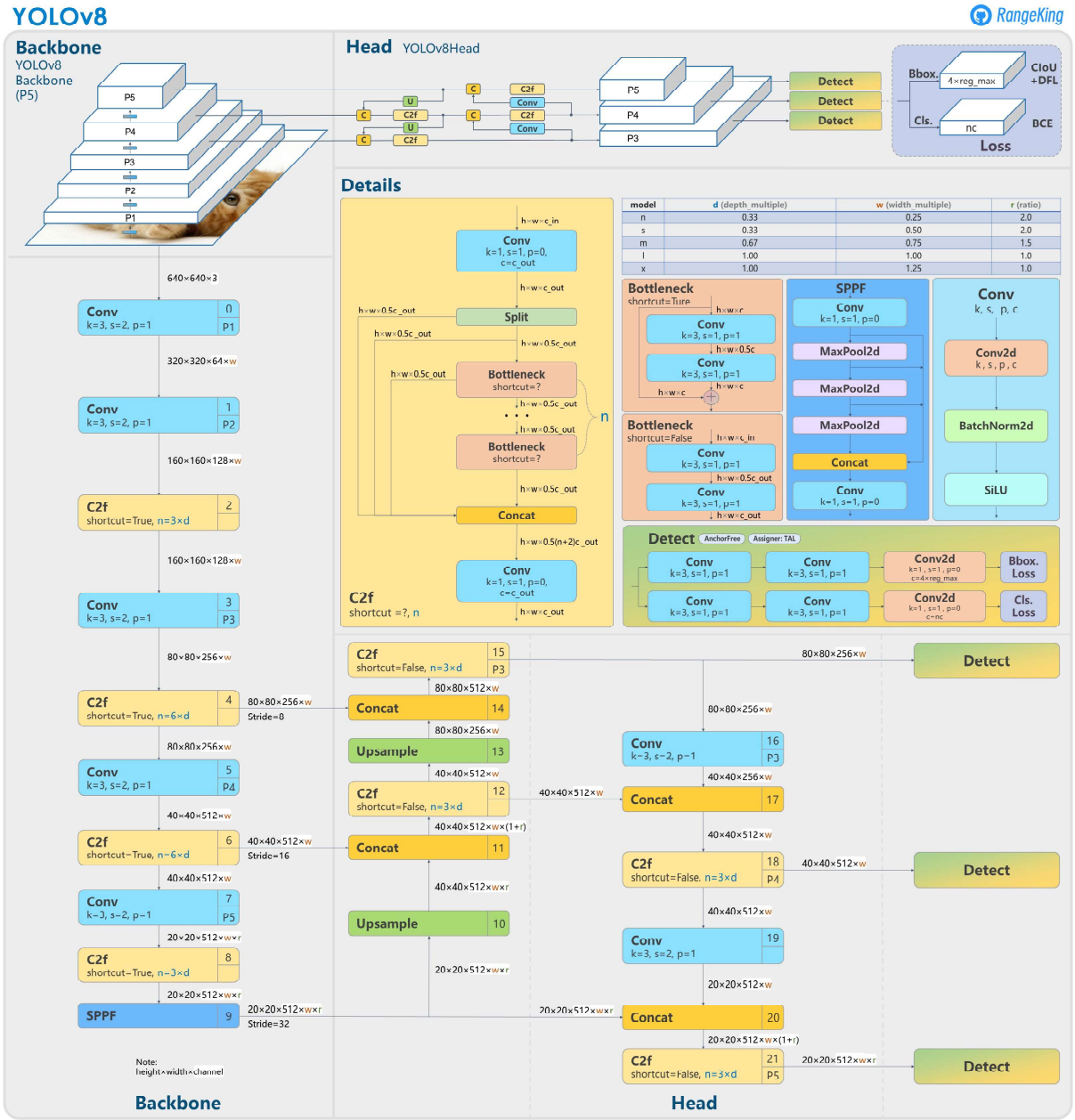


Figure 4.24: Yolo v8 architecture overview [118]

For the two-stage detector, Faster R-CNN [60], the X101-FPN variant was employed, which will be referred to as Faster model in the following section, for increased readability. The X101-FPN is known to achieve superior results compared to other available variants, supported by the comparison presented in Table 4.9.

Table 4.9: Faster R-CNN model variants performance comparison [119]

Name	LR sched	Train time (s/iter)	Inference time (s/im)	Train mem (GB)	Box AP
R50-C4	1x	0.551	0.102	4.8	35.7
R50-DC5	1x	0.380	0.068	5.0	37.3
R50-FPN	1x	0.210	0.038	3.0	37.9
R50-C4	3x	0.543	0.104	4.8	38.4
R50-DC5	3x	0.378	0.070	5.0	39.0
R50-FPN	3x	0.209	0.038	3.0	40.2
R101-C4	3x	0.619	0.139	5.9	41.1
R101-DC5	3x	0.452	0.086	6.1	40.6
R101-FPN	3x	0.286	0.051	4.1	42.0
X101-FPN	**3x**	**0.638**	**0.098**	**6.7**	**43.0**

This model combines 101 layers of the ResNetX [120] backbone with standard convolutional and fully connected heads for mask and bounding box prediction, respectively. This predictor head architecture is known for achieving the best speed/accuracy tradeoff. The model was initialized with the weights of the ResNeXt-101-32x8d model, available from *Model Zoo* repository [119].

In this implementation the default input size configuration was used, constrained by computational resources, that resizes the shortest edge of the input image to a random value chosen from a range of values (640, 672, 704, 736, 768, 800), ensuring that the resized longest edge does not exceed the maximum size of 1333 pixels. The default configuration for data augmentation was also used, which randomly flips the input image horizontally with a probability of 0.5.

Since it is a two-stage detector, the batch size configuration was set to 4 images per batch, with a batch of 64 region proposals generated per image during training.

The learning rate was scheduled to start at 0.01 and decrease by a factor of 10 in each stage of 1000 iterations, for a total of 5000 iterations.

In addition to the Detectron2 official documentation [119], one of the references for successfully conducting the Faster R-CNN implementation on a custom dataset was the work of Mon Kira published on kaggle [114].

The ResNetX architecture [120] is an efficient extension of the original residual networks, designed to mitigate the problem of the vanishing gradient, by reducing path lengths further through the incorporation of two structural features. One is a larger proportion of shorter paths compared to ResNet, which improves the direct flow of information through the entire network. Another structural feature is a high degree of disorder, illustrated by Directed Acyclic Graphs (DAG), presented in the bottom of Figure 4.25 which means nodes tend to connect to other nodes with different levels, further improving the multi-scale representation of the model.

In addition to the existing dimensions of *depth*, *width*, and *cardinality* introduced in ResNeXt [121], the ResNetX architecture introduces a new dimension, known as *'fold depth'*. The deeper the fold, the larger the number of residual connections or 'direct' chains between layers in the network. This fusion of a larger number of feature maps with different receptive fields improves the model's multiscale representation ability.

The top of Figure 4.25 presents diagrams of ResNetX architectures for folds with depths of 1 (equivalent to the original ResNet), 2, and 3, respectively. The data flow starts from the external input image, where plus signs represent the summation of image data. Dashed lines represent identity functions, while solid lines labeled [F] represent functions of non-linear transformations.

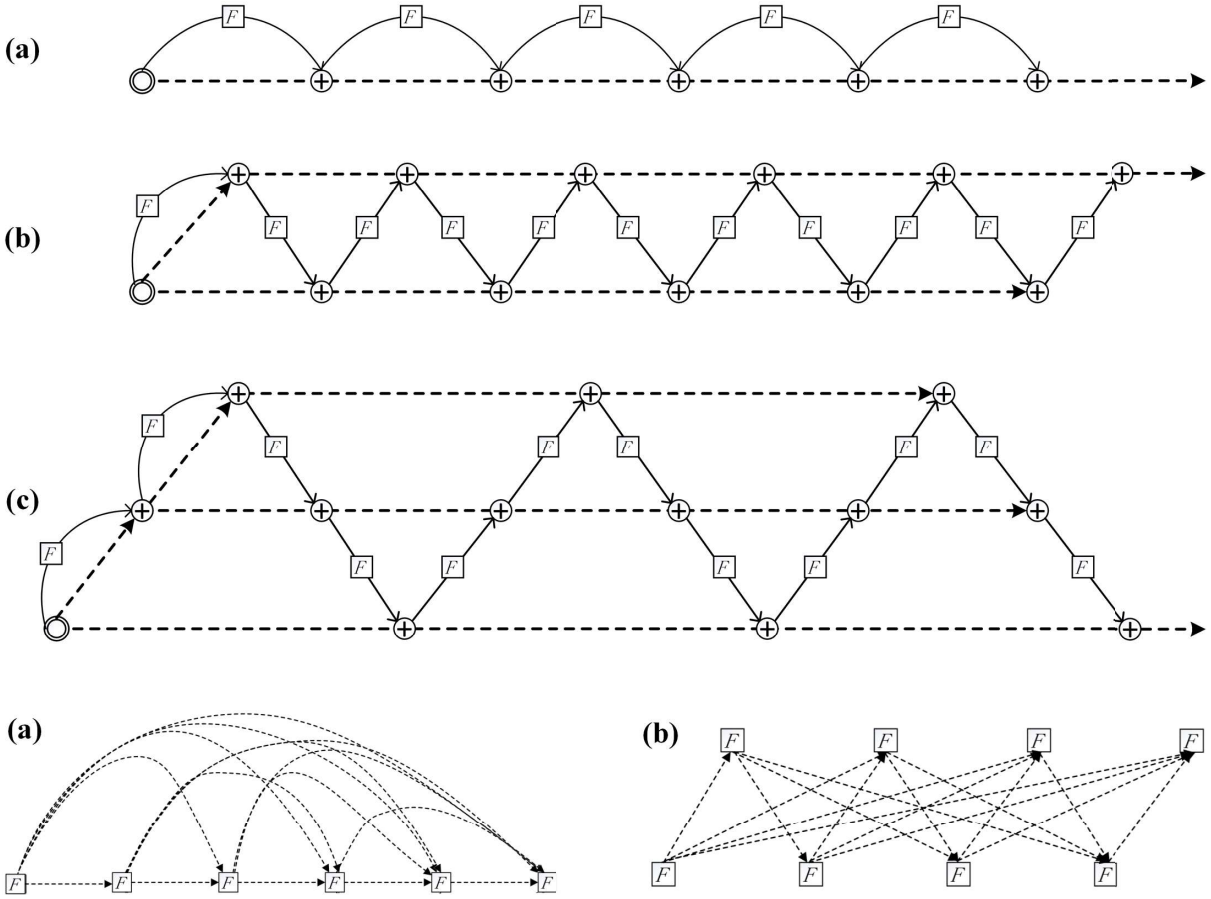


Figure 4.25: Diagrams of ResNet, ResNetX (t=2), and (t=3) network architectures [120]

4.3.4 Results

This section presents quantitative evaluations, from the YOLO and Faster implementations, using standard metrics, including the mean Average Precision (mAP) and the mean Average Recall (mAR), across typical Intersection over Union (IoU) thresholds.

The results presented were automatically obtained by inferring the validation dataset split, as it contains annotations. Both of the testing datasets are not annotated, so they were used only for qualitative evaluation in the following section.

The information in Figure 4.26 reveals that the YOLO model has successfully converged along the training iterations. Despite a consistent decrease in training loss throughout the epochs, there was no improvement in the validation loss after epoch 56, suggesting the starting of overfitting. Therefore, training was stopped early at epoch 71, since it was configured to stop after 15 epochs with no improvement in the validation loss.

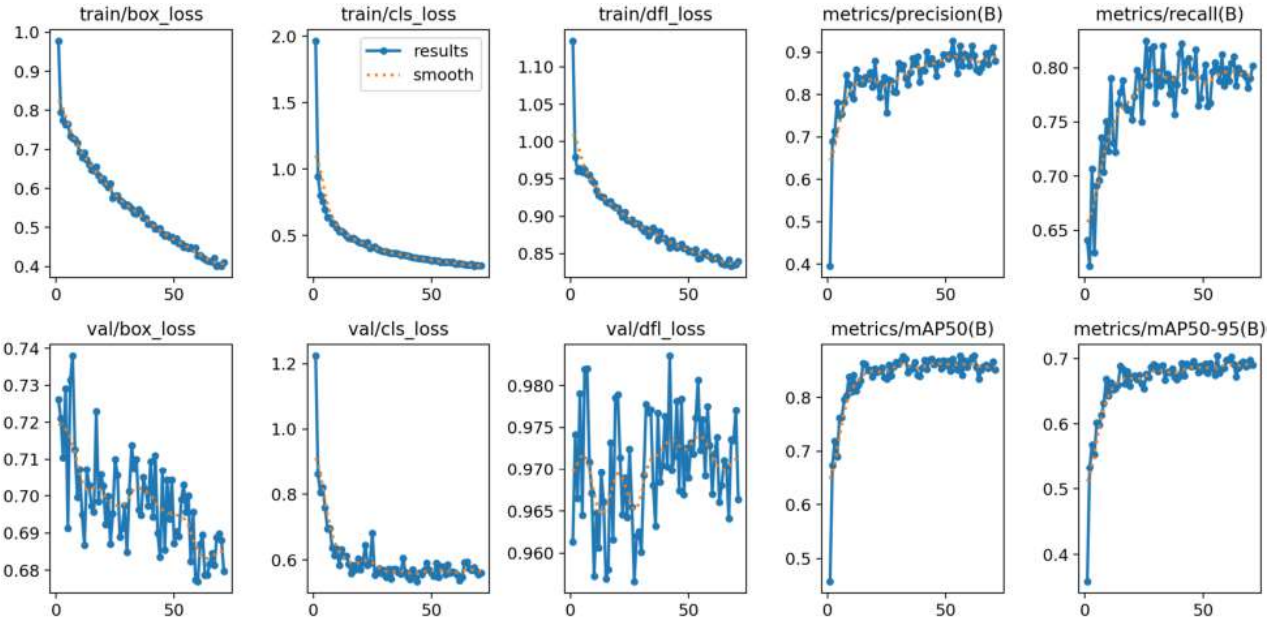


Figure 4.26: Evolution of losses and metrics during YOLOv8s model training

Figure 4.27 presents the curves of Precision, Recall, and F1-score as functions of the confidence threshold level, along with the precision-recall curve at 0.50 IoU. Additionally, it includes in the legend annotations the curve maximum point.

The *good* class has the lowest precision but the highest recall. In contrast, the other classes have lower recall but high precision. For instance, since the *broken* class represents less than 10% of the instances but has highly distinctive features, this might indicate that the model is highly conservative in its predictions.

However, in a scenario with multiple images of the same asset, this might not be a significant problem, considering that there will always be an image showing the defective features in an optimal orientation.

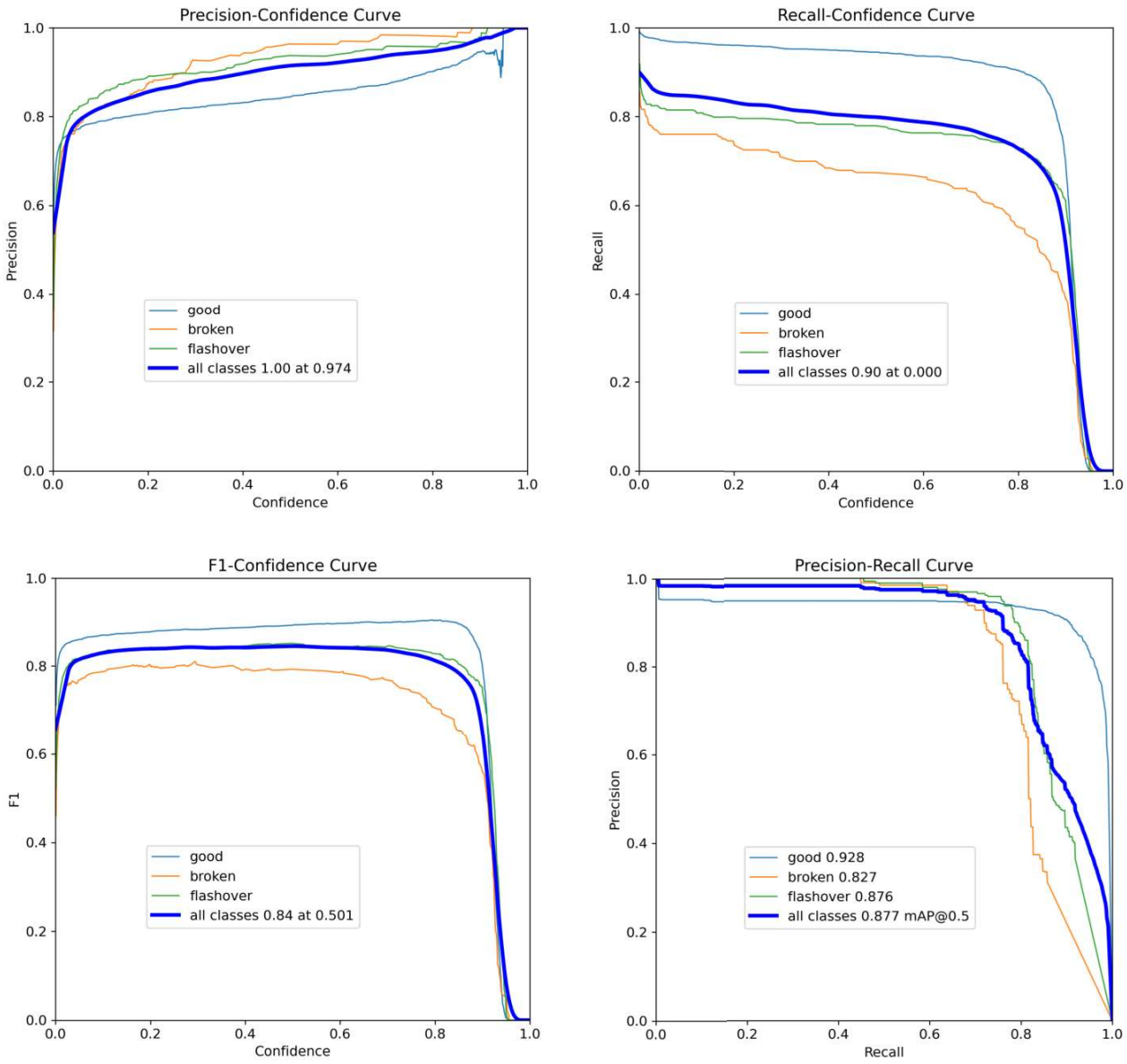


Figure 4.27: Main classification metric curves on the trained YOLOv8s model

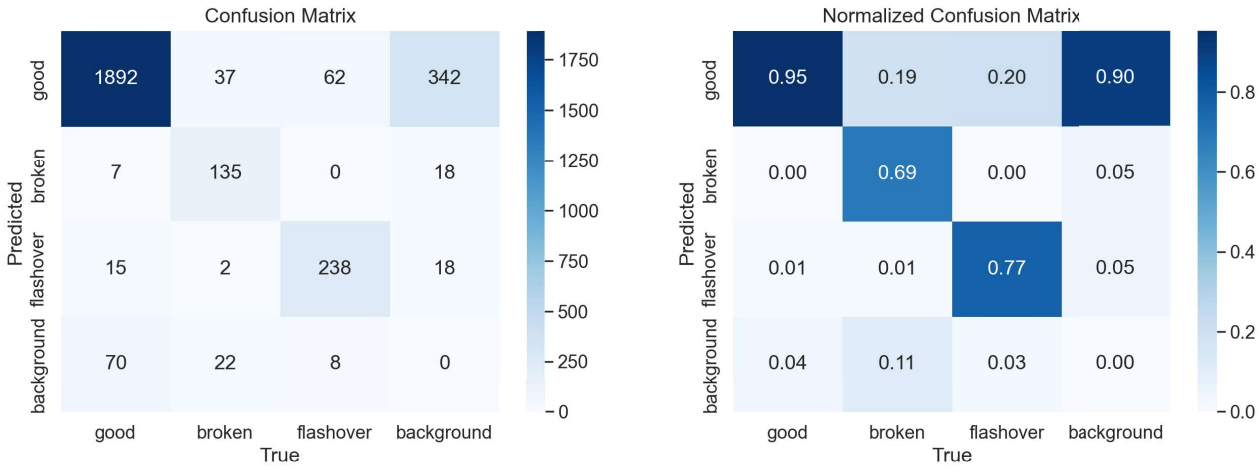


Figure 4.28: Absolute and normalized confusion matrices on the trained YOLOv8s model

Observing Figure 4.28, which illustrates the confusion matrix with absolute and normalized values, respectively, both computed at a default confidence threshold of 0.25, it is evident that several instances, particularly in the 'good' class, have several predictions where the ground truth label is the background. One justification for this metric is that not all instances are annotated.

In fact, upon closer inspection of some images, such as the one presented in Figure 4.29, it is noteworthy that disc insulators cropped by the edge of the image or *good* insulator discs adjacent to broken ones do not have annotations. This decision potentially arose from the overlapping context of the bounding boxes. Another relevant information, which aligns with the insights from the precision and recall curves, is that about 20% of the defective assets are being predicted as functional ones.

Figure 4.29: Example of missing annotations in the annotated dataset IDID

The standard performance metrics, easy accessible from Ultralytics APIs are presented in Table 4.10. The noticeable performance decay from the mAP50 to the mAP50-95 indicates that the bounding box alignment might be suboptimal.

However, to rely heavily on this metric it is imperative that geometric precision of the annotations are considered trustworthy.

Table 4.10: Summary of class-wise validation performance on YOLOv8s

Class	Instances	Box P	Box R	mAP50	mAP50-95
all	2488	0.915	0.799	0.879	0.706
good	1984	0.844	0.945	0.929	0.789
broken	196	0.962	0.673	0.827	0.609
flashover	308	0.937	0.777	0.880	0.719

Observing Figure 4.30 can be useful for understanding the classification performance and dataset characteristics. It provides information about the correlation between the different labels, in this case, the bounding box values (x, y, w, h), which helps identify relationships or similarities between them.

The position (x) and (y) of the objects follows a symmetrical normal distribution, with the mean value close to the image center. However, the width (w) and height (h) have an asymmetrical distribution with positive skewness. Additionally, the relationship between width and height is approximately a direct proportionality.

The relation between horizontal (x) and vertical (y) position indicates that a significant portion of the insulator strings are vertical and centered in the image, which is why the (y) distribution has a higher standard deviation that the (x) distribution.

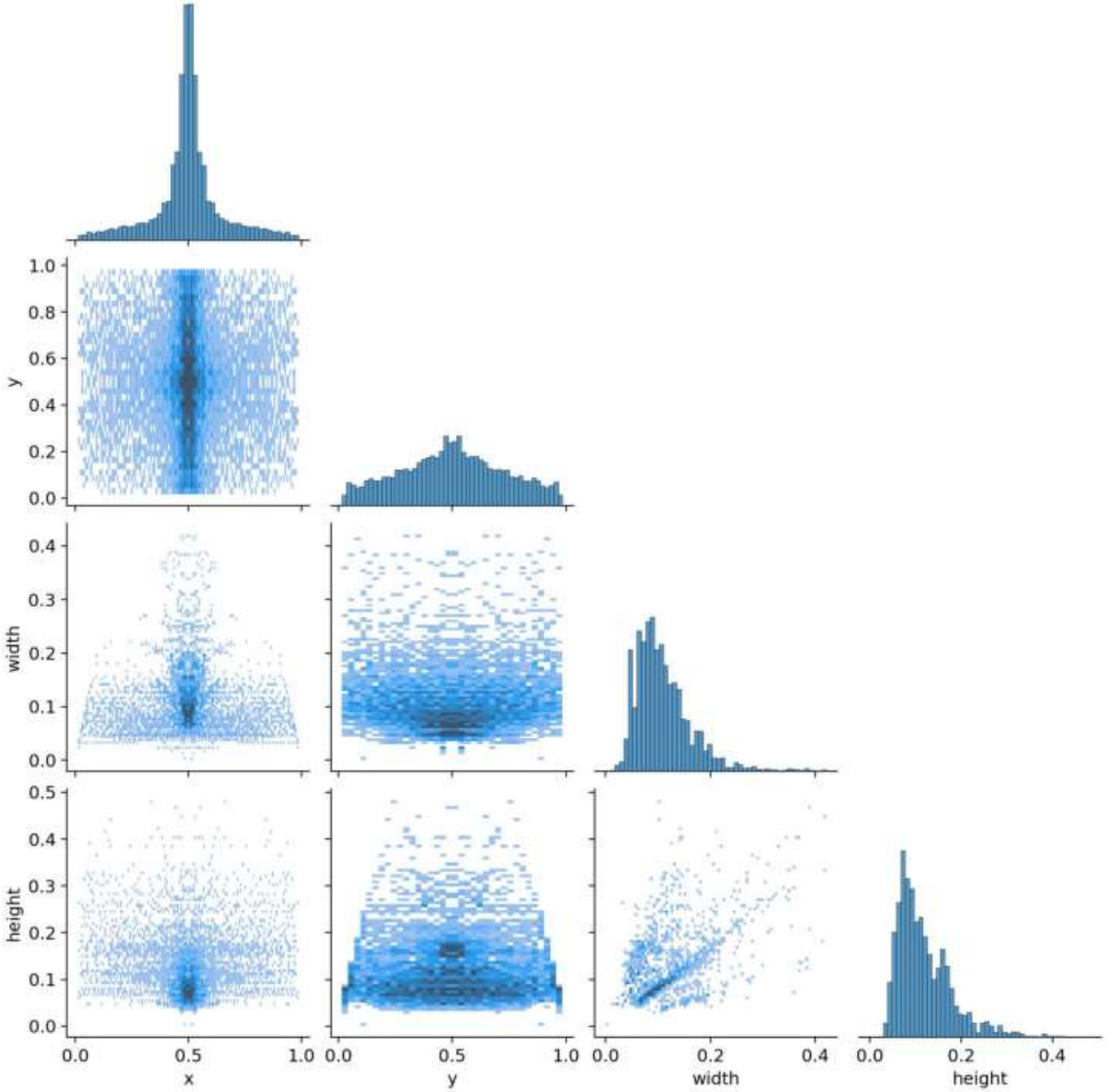


Figure 4.30: Distribution and correlation of the bounding box in the trained YOLOv8s model

In the Faster model training iterations, due to the small batch size, the loss keeps going up and down in the last stages. This happens because, in every iteration, the model tries to adjust to a small portion of the data.

Class-wise analysis revealed high precision for the common class, namely *good* insulator, while performance was slightly lower for less frequent classes like *broken* and *flashover*. As shown in Table 4.11, the average precision is relatively high, and higher for mAP at 0.50 IoU threshold limits in all classes.

Table 4.11: Summary of class-wise validation performance on Faster R-CNN X101-FPN

Metric	IoU	max Dets	Class			
			All	Good	Broken	Flash
mAP	@0.50	100	0.872	0.943	0.830	0.842
	@0.75	100	0.783	0.932	0.586	0.831
	@0.50:0.95	100	0.672	0.777	0.558	0.682
mAR	@0.50:0.95	100	0.761	0.847	0.664	0.772
	@0.50:0.95	10	0.714	0.723	0.664	0.756
	@0.50:0.95	1	0.283	0.099	0.483	0.266

The significant drop in mAP at 0.75 IoU in the *broken* class is probably related to the fact that sharp edges lead to different bounding box regression results. As mentioned earlier, the trustworthiness of these metrics is closely linked to the trustworthiness of the annotations, which requires a deeper analysis of the dataset.

In this scenario of multiple instances of an object class in an image, setting the maximum detections to 1 severely limits the model's ability to recall all instances of objects in the image, since the model predicts only one bounding box, the one that have the top score prediction, for each object class present in an image, missing the others.

By looking at recall in the initial detection, it is not surprising that the *broken* class has the higher value and the *good* class the lowest, since they are the lower and the higher representative class, respectively. In the defective classes, within the first 10 detections (in the same image), the model identifies a significant portion of all the objects it will eventually detect. An exception to this trend is observed in the *good* class, which still experiences a significant increase in recall after the 10^{th} detection. This is primarily because some images contain more than 10 good insulator shells.

In other works [114], where the same dataset was used, higher values for the metrics are presented, yet, using a random validation split in this specific case could compromise the obtained results due to the presence of mirrored data.

A direct comparison of the validation results obtained in this work using the YOLO and Faster models, as presented in Table 4.12, reveals that YOLOv8 has surpassed Faster R-CNN in the most demanding metric (mAP50-95) across all classes.

Table 4.12: Summary of class-wise validation mAP@0.50:0.95 IoU comparison

mAP	**Model**	**Class**			
		All	Good	Broken	Flashover
@0.50:0.95	YOLOv8	**0.706**	**0.789**	**0.609**	**0.719**
	Faster R-CNN	0.672	0.777	0.558	0.682

However, it would be unfair to say that the YOLOv8 model always performs better than the Faster R-CNN model. In fact, a few examples where YOLO performed well while Faster did not, and vice-versa, are further presented in Figure 4.35.

To compare the validation results obtained in this work with those obtained in other studies that used the same dataset, a more common metric was also used, the mAP at 0.50 IoU, as other metrics were not available in the assessed literature.

Table 4.13 summarizes the validation results of the mean Average Precision at the 0.50 Intersection over Union threshold level in a class-wise comparison.

Table 4.13: Summary of class-wise validation mAP@0.50 IoU comparison

mAP	**Model**	**Class**			
		All	Good	Broken	Flashover
@0.50	YOLOv8s	**0.879**	0.929	0.827	**0.880**
	Faster R-CNN	0.872	**0.943**	**0.830**	0.842
	ML YOLOv5	0.776	0.834	0.748	0.745
	YOLOv5	0.327	0.760	0.000	0.220

This summary highlights the marginal superiority of the Faster model in the *good* and *broken* classes, indicating that while the YOLO model may achieve better bounding box regression accuracy, the Faster model may exhibit higher classification accuracy.

Additionally, it can be observed that the results for the models implemented in this study outperform others found in the literature, more precisely, the results obtained by Laya Das et al. [122], that utilized YOLOv5, and the results obtained by using a new variant named ML YOLOv5, proposed by Tong Wang et al [123].

4.3.5 Evaluation

By observing the images in the testing dataset, which share similar characteristics to the annotated dataset, it is clear that both of the models (YOLO in Figure 4.31 and the Faster R-CNN in Figure 4.32), successfully predicted the considered instances, identifying both defective insulator disks and functional ones in the majority of the images. A broader sample of testing inference images is presented in Appendix C.

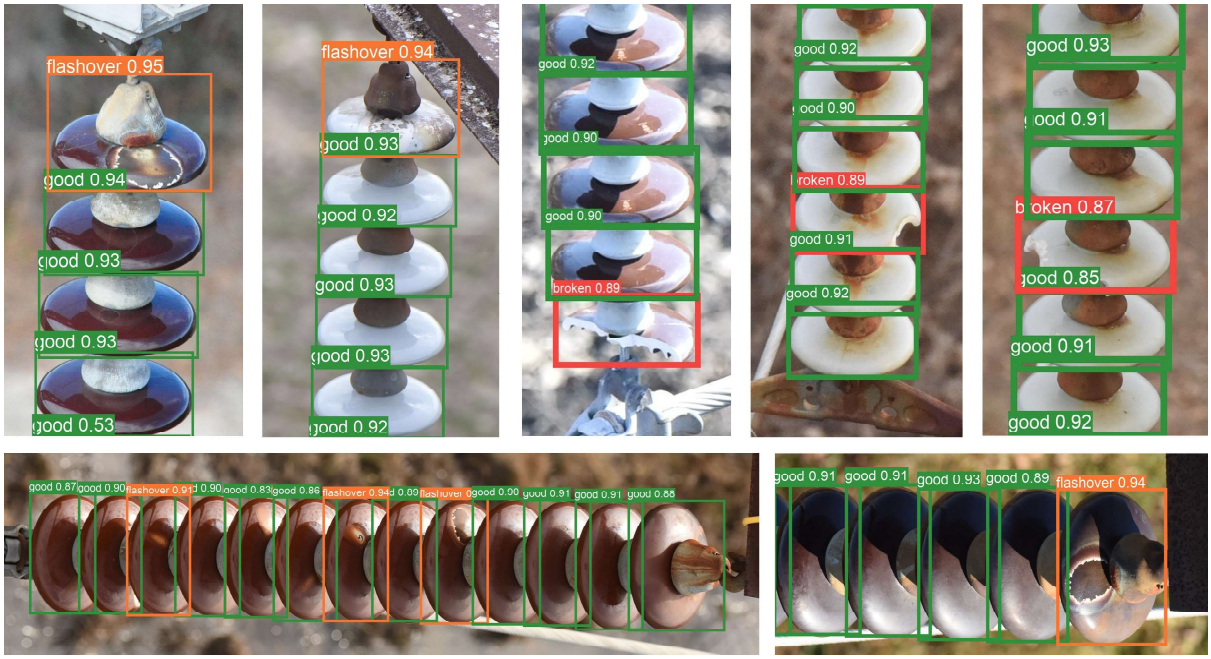


Figure 4.31: Correct predictions on inference test of IDID using YOLOv8s

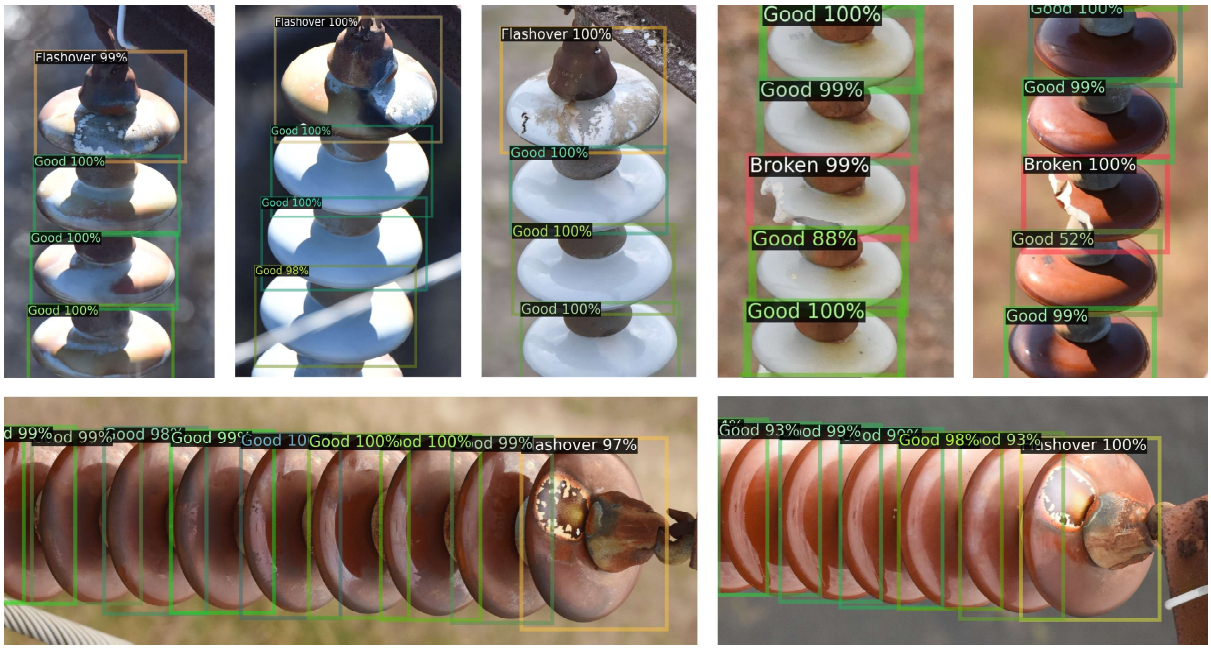


Figure 4.32: Correct predictions on inference test of IDID using Faster R-CNN X101-FPN

By assessing some examples of miss-predictions, it is possible to find patterns in the characteristics of the images and the objects wrongly detected or classified.

The 1st image in Figure 4.33 indicates that the YOLO model has difficulties identifying the actual texture of a broken insulator disk. The 2nd and 3rd images show that insulator disks contaminated with bird droppings alike are easily confused with flashover defects. In the 4th image, a defective insulator disk is correctly identified but with the wrong defect class. The presence of a power line in the foreground partially obstructing the object's view can impact the class prediction, as illustrated in the 5th image. In the 6th image is an object with multiple detections, having similar bounding boxes but different classification results. The 7th and 8th images show a flashover prediction due to unusual textures. Defocused objects in the 9th image are considered background, mainly because the training data consists of close planes with the focal point on the object's surface. The last image suggests that elliptical shapes that are not disc insulators can be misclassified as defective ones.

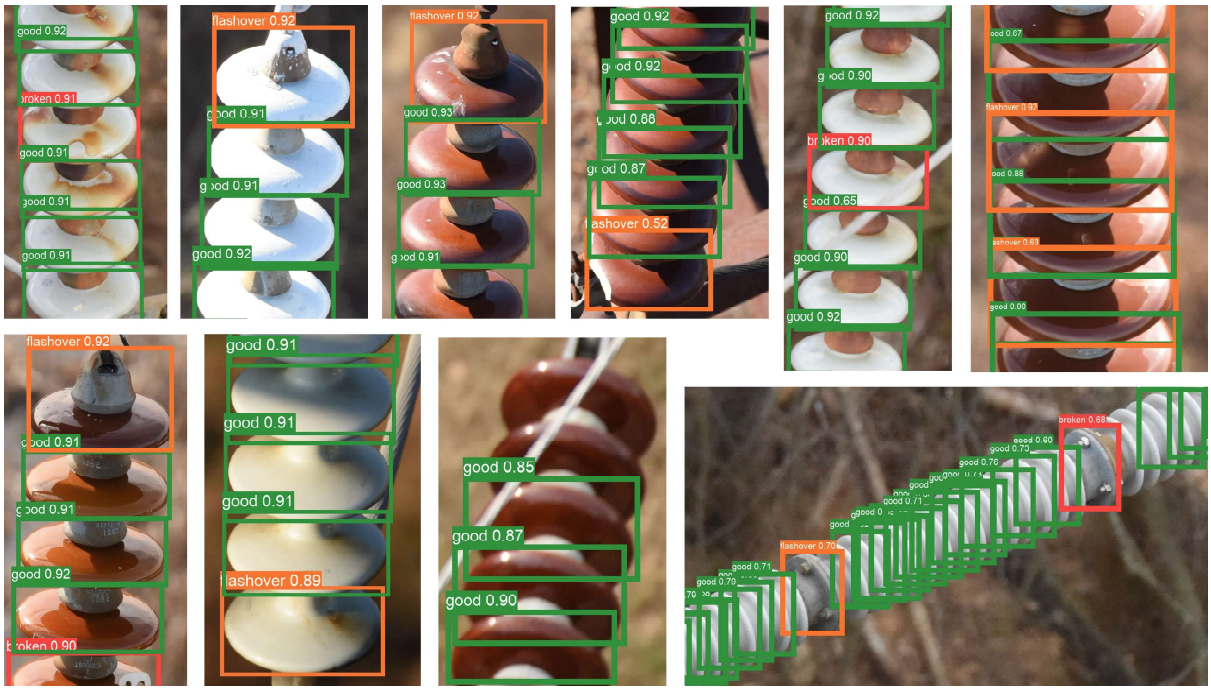


Figure 4.33: Miss-predictions on inference test of IDID using YOLOv8s

The first 3 images in Figure 4.34 suggest that the Faster model struggle to detect a broken insulator disk if it does not have a brighter, sharp edge visible. The second group of 2 images exemplify that *good* insulators next to *broken* ones are often not detected. This trend aligns with the annotation practices adopted. Illustrated by the 6th and 7th images are objects with multiple detections, having similar bounding box but different classification results. If the main goal is identifying images with defects, that issue may not be a problem. Next, the misclassification of a *good* insulator as a *broken* one suggests that the power line in the foreground might have influenced the class prediction. The

model is clearly not well-fitted yet for evening light conditions and prominent shadows due to a lack of training examples (e.g. 8th image). Lastly, translucent glass insulators cannot be correctly detected, due to the absence of training examples.

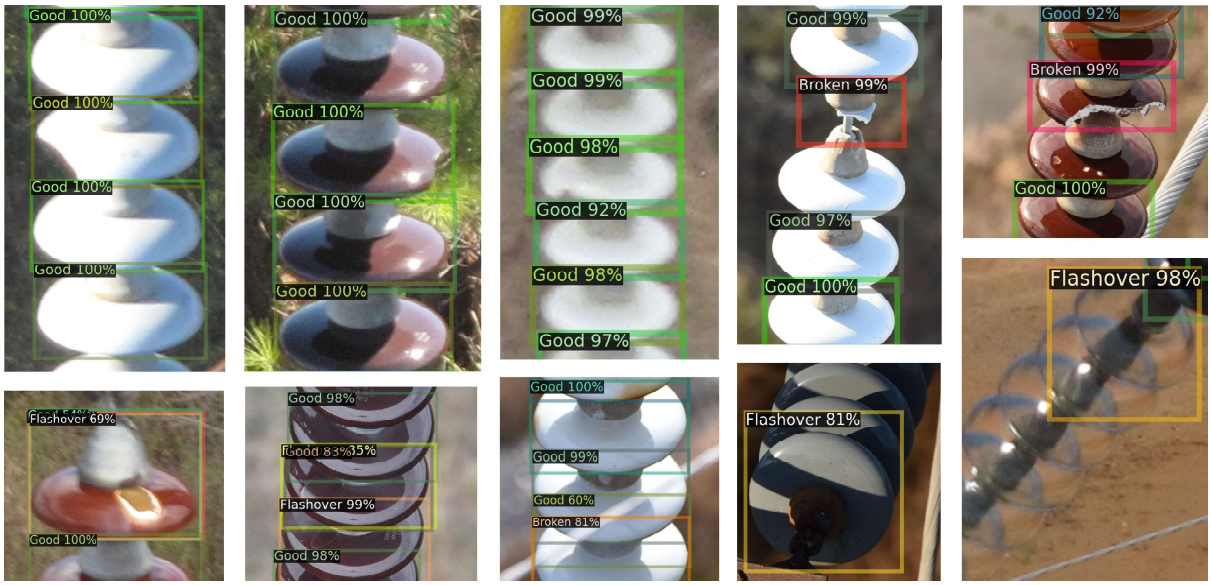


Figure 4.34: Miss-predictions on inference test of IDID using FasterR-CNN X101-FPN

A direct comparison between the inference results reveals that both models outperform each other under different conditions, as illustrated in Figure 4.35, where the first line shows YOLOv8 predictions and the last line shows Faster R-CNN predictions.

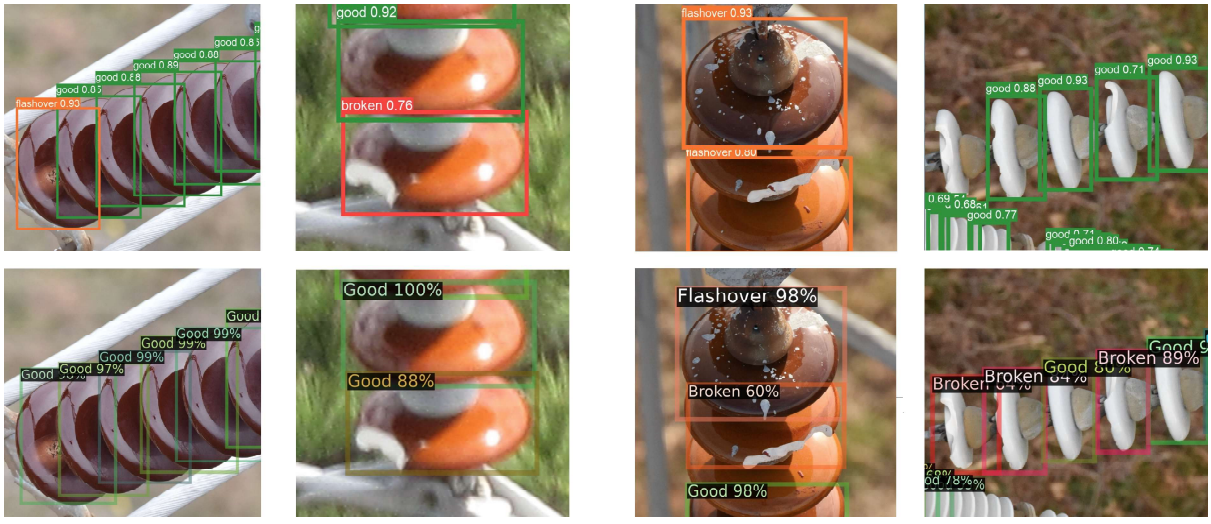


Figure 4.35: Examples of miss predictions exclusively in one of the models

Another non-annotated testing dataset, owned and managed by EDP LabElec, was made available at the final stage of this work. Given the time constraints of this project, it was not possible to perform enough annotations to create a large-scale training dataset that enabled a fine-tuning process. However, some images existing in this dataset were

subjected to inference by both models, validating the potential of the approach conducted in this work in detecting disc insulator objects and classifying them as functional or defective. Figure 4.36 and Figure 4.37 illustrate cropped images resulting from EDP LabElec data inference on the YOLO and Faster models respectively.

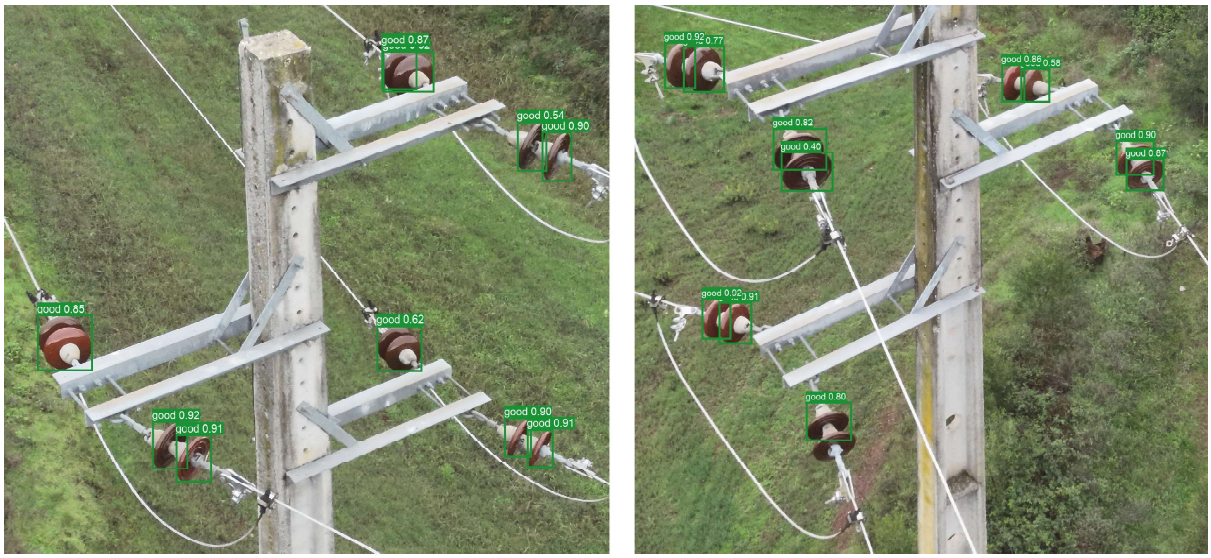


Figure 4.36: Inference test result of EDP LabElec data using YOLOv8s

Figure 4.37: Inference test result of EDP LabElec data using Faster R-CNN X101-FPN

This test inference on a dataset with different characteristics from the one used for training revealed that the trained YOLO model achieved better generalization capabilities. While the Faster model struggled to make valuable detections on images with blur objects, primarily because the training data consists of close planes with the focal point on the object's surface.

5 Conclusion

The development of this project complemented the knowledge acquired through various Master's program courses, particularly, Maintenance and Quality Control, and Computer Vision and Multimedia.

This project provided an opportunity for an entry point into cutting-edge technologies, specifically in the fields of neural networks and machine learning. It also strengthened software development skills, expanded the acquired toolbox, and established proficiency in Python, a high-level object-oriented programming language only used at an introductory level in the Electrical Engineering program.

Practical experiments, supplemented by iterative adjustments based on feedback, stimulated intuition for tuning deep learning hyperparameters. Such skills could be valuable for pursuing a career in data engineering.

But more importantly, all the informal study and research motivated by this project led to the forefront of technological advancements in computer science.

5.1 Summary

The conducted research included the fundamental concepts of asset management and maintenance practices, the scientific principles behind the photovoltaic effect and the causes and symptoms of failures in PV modules and high-voltage ceramic insulators.

It involved reviewing computer vision applications and the digital image processing techniques that enable the understanding of artificial intelligence supported by deep learning models. The related work reviews on electrical assets inspection, namely, PV power generation and T&D systems, ensured a front-line beginning for the experimental work. In addition, the study on supporting frameworks and tools for developing deep learning models, resulted in a valuable compilation of information tailored for electrical engineers interested in developing automated visual inspection systems for electrical assets.

In terms of specific automated inspection applications, images of PV Cells, PV Modules and HV Insulators were used to investigate the performance of deep-learning algorithms for binary and multi-classs classification, and also for object detection tasks.

A classic network, VGG-19, was employed to identify the operational conditions of PV Cells by analyzing electroluminescence images sourced from a public labeled dataset.

The classification results yielded an AUC of 0.95, 0.94 and 0.90, respectively for polycrystalline, monocrystalline, and both types mixed dataset, achieving slight improvements compared to the results reported in the literature reference study.

For PV Modules, multiple shallow CNN architectures, to analyze defects within 12 classes of UAV-collected thermographic imagery sourced from a public labeled dataset, were tested and evaluated. The classification results yielded Precision, Recall and F1-score of 0.87, 0.86 and 0.86, respectively, demonstrating the practical value of the presented automated approach.

Defects in HV Insulators were detected utilizing SOTA deep learning models to process visible light images sourced from a publicly available dataset, comprising 1600 images with over 17000 annotations of insulators discs. Both models presented satisfactory performance results, with a mean average precision (mAP@50), for the three considered classes (*Good* insulator; *Broken* disc; and *Flashover* shell), of 87,9% and 87,2% on the YOLOv8s and Faster R-CNN X101-FPN respectively.

To understand the perspective of a Distribution System Operator (DSO) and to contribute to the industry efforts in this area, a collaboration agreement was established with EDP LabElec, providing important professional insights and the opportunity to experiment with real-world data.

5.2 Challenges

Some theoretical lessons were empirically reinforced, for instance, highlighting the underrated importance of data quality in developing robust deep learning models. While diversified data promotes generalization, outliers prevent an optimal fitting achievement. This was noticed in the application for multi-classification of PV modules. When evaluating the inference results, doubtful labeled images emerged has a reason for lowe performance metrics across various training attempts. To confirm this hypothesis, a meticulous data analysis and selection was conducted, following the guidelines outlined in the dataset documentation. Specifically, focusing on identifying image patterns associated with each defect class.

The conducted Grad-CAM evaluation of the obtained results helped on identifying occasions when the feature extraction layers failed to learn meaningful patterns. This encouraged retraining the model despite the overall satisfactory metrics' performance.

A major challenge faced during this project was the limitation of computational resources, particularly when dealing with complex architectures like YOLOv8x (extra large), especially in the context of high-resolution images. Although cloud-based platforms offer powerful GPUs and TPUs for development of deep learning models, access under free accounts is often limited, typically to maximum sessions of 12 hours.

These constraints affected the obtained results, limiting the experiments to low resolution resized images and lighter models with few parameters such as YOLOv8s (small). While these models may sacrifice some accuracy, they significantly improve training and inference performance.

5.3 Future Work

A future direction for improving the value of the *PV Cells* model is to include, in the dataset, information about the power generated under reference conditions. Addressing this problem as a regression task, instead of binary classification, would allow for the prediction of efficiency degradation results and avoid subjective classification assessments.

To increase the performance of the *PV Modules* model, different datasets should be used to train it, in order to upgrade its adaptability. A more advanced model might be obtained by merging thermography analysis with other sensors such as RGB cameras and V-I instruments, to extract patterns between the electrical measurements and the PV module degradation, in order to perform real-time inspections with a lower cost.

Both of these PV applications have the potential to evolve into predictive maintenance tools. By incorporating dense time-series data containing operational period values into the dataset, it becomes possible to develop a model capable of predicting the expected efficiency curve during its lifetime based on the external symptoms of its current degradation state, by identifying non-linear patterns in the degradation process.

The inspections of *HV Insulators* can be expanded to other T&D assets, conditioned by the elaboration of annotated datasets containing the respective objects. Future improvement proposals regarding the model architecture comprehend the implementation of a pipeline of two object detectors, where the first focus on locating the insulator string in a subsampled image, and the second use the bounding box prediction to crop the high-resolution original image, benefiting from the information-data improved ratio. In the UAV data collection process, embedded computer vision systems can be used to validate the quantity and quality of the data to be stored.

From a maintenance management perspective, this application could be even more comprehensive and valuable by providing alternative ordered lists of the data inference outcomes, such as failure risk or replacement costs, considering that this information can be fed to the model in a supervised learning fashion.

After this successful investigation, the short-term plan is to consolidate its expansion into a broader project within an existing collaboration with a Distribution System Operator (DSO), namely, EDP LabElec. The aim is to create a high quality large annotated dataset, optimize the hyper-parameter selection, and fine-tuning a powerful variant of the YOLOv8 model to achieve industry-leading results.

Bibliography

[1] "Executive summary – electricity grids and secure energy transitions – analysis," https://www.iea.org/reports/electricity-grids-and-secure-energy-transitions/executive-summary, accessed: 2023-12-12.

[2] https://www.nrel.gov/docs/fy22osti/83586.pdf, accessed: 2023-12-12.

[3] https://www.researchgate.net/publication/335923501_Deterioration_of_Porcelain_Insulators_Utilized_in_Overhead_Transmission_Lines_A_Review, accessed: 2023-12-12.

[4] AISPECO, "Power lines inspection methods," https://www.linkedin.com/pulse/power-lines-inspection-methods-aispeco/, Feb. 2022, accessed: 2023-12-12.

[5] Z. Ma, L. Zhou, and W. Sheng, "Analysis of the new asset management standard ISO 55000 and PAS 55," in *2014 China International Conference on Electricity Distribution (CICED)*, Sep. 2014, pp. 1668–1674, iSSN: 2161-749X. [Online]. Available: https://ieeexplore.ieee.org/document/6991990

[6] E. summary:, "How ISO 55000 can help transform utility operations through better asset management," https://www.aveva.com/content/dam/aveva/documents/white-papers/WhitePaper_AVEVA_TandDISO55000_21-06.pdf, accessed: 2023-11-4.

[7] "Assessing PV Module Reliability | Greentech Renewables," Nov. 2013. [Online]. Available: https://www.greentechrenewables.com/article/assessing-pv-module-reliability

[8] Q. Wen, Z. Luo, R. Chen, Y. Yang, and G. Li, "Deep Learning Approaches on Defect Detection in High Resolution Aerial Images of Insulators," *Sensors*, vol. 21, no. 4, p. 1033, Feb. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1033

[9] F. Zhou, G. Wen, Y. Ma, H. Geng, R. Huang, L. Pei, W. Yu, L. Chu, and R. Qiu, "A Comprehensive Survey for Deep-Learning-Based Abnormality Detection in Smart Grids with Multimodal Image Data," *Applied Sciences*, vol. 12, no. 11, p. 5336, May 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/11/5336

[10] "UV & IR Imaging of Defects in MV/HV Components." [Online]. Available: https://www.inmr.com/uv-ir-imaging-of-defects-in-hv-components/

[11] E. Engel and N. Engel, "A Review on Machine Learning Applications for Solar Plants," *Sensors*, vol. 22, no. 23, p. 9060, Jan. 2022.

[12] S. Europe, "Global market outlook for solar power 2023 - 2027," Tech. Rep., 2022.

[13] M. K. *et al.*, "Review of failures of photovoltaic modules," IEA International Energy Agency, Tech. Rep., 01 2014.

[14] L. Bommes, C. Buerhop-Lutz, T. Pickel, J. Hauch, C. Brabec, and I. Marius Peters, "Georeferencing of photovoltaic modules from aerial infrared videos using structure-from-motion," *Progress in Photovoltaics: Research and Applications*, vol. 30, no. 9, pp. 1122–1135, Sep. 2022.

[15] "Meet the team working to keep the electric grid energized - noteworthy AI," https://www.noteworthy.ai/about, accessed: 2023-12-9.

[16] V. N. Nguyen, R. Jenssen, and D. Roverso, "Intelligent Monitoring and Inspection of Power Line Components Powered by UAVs and Deep Learning," *IEEE Power and Energy Technology Systems Journal*, vol. 6, no. 1, pp. 11–21, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8610301/

[17] "Raptor maps - resources - 2023 global solar report," https://raptormaps.com/resources/2023-global-solar-report, accessed: 2023-11-15.

[18] V. inspections., "Assessing the value and efficiency of virtual inspections," https://www.gridvision.com/wp-content/uploads/2023/09/Assessing-the-value-and-efficiency-of-virtual-inspections.-A-comparative-study-of-foot-patrol-and-virtual-inspections..pdf.pdf.

[19] "Epri - power delivery inspection imagery data set," Electric Power Research Institute, IEEE Dataport. [Online]. Available: https://www.epri.com/thought-leadership/artificial-intelligence/data-sets/3QmA0kn43RnN56uAlbkMHG

[20] "EPRI Electric Transmission Imagery." [Online]. Available: https://kaggle.com/competitions/electric-transmission-imagery

[21] M. Aghaei, A. Fairbrother, A. Gok, S. Ahmad, S. Kazim, K. Lobato, G. Oreski, A. Reinders, J. Schmitz, M. Theelen, P. Yilmaz, and J. Kettle, "Review of degradation and failure phenomena in photovoltaic modules," *Renewable and Sustainable Energy Reviews*, vol. 159, p. 112160, May 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032122000880

[22] SciToons, "How do solar cells work?" Apr. 2018.

[23] T. Sun, H. Xing, S. Cao, Y. Zhang, S. Fan, and P. Liu, "A novel detection method for hot spots of photovoltaic (PV) panels using improved anchors and prediction heads of YOLOv5 network," *Energy Reports*, vol. 8, pp. 1219–1229, Nov. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S235248472201575X

[24] J. Starzyński, P. Zawadzki, and D. Harańczyk, "Machine Learning in Solar Plants Inspection Automation," *Energies*, vol. 15, no. 16, p. 5966, Jan. 2022.

[25] J. P. Holtzhausen, "HIGH VOLTAGE INSULATORS," https://web.archive.org/web/20140514000839/http://www.idc-online.com/technical_references/pdfs/electrical_engineering/highvoltage.pdf, accessed: 2023-12-6.

[26] "Insulators and technical services for electric, utility, OEM customers," https://www.victorinsulators.com/, Mar. 2017, accessed: 2023-12-6.

[27] E. Iq, "How suspension disc insulator are connected !! 11KV suspension disc insulator !! string efficiency," Apr. 2023.

[28] T. Kim, S. Sanyal, J.-B. Koo, J.-A. Son, I.-H. Choi, and J. Yi, "Analysis of long-term deterioration characteristics of high voltage insulators," *Appl. Sci. (Basel)*, vol. 10, no. 1, p. 123, 2019.

[29] polluted conditions – Part 1: Definitions and general principles, "iteh STANDARD PREVIEW," https://cdn.standards.iteh.ai/samples/13921/bffb9fe6f30448768ff390c9e29cd05e/IEC-TS-60815-1-2008.pdf, 2008, accessed: 2023-12-6.

[30] J. Looms, *Insulators for High Voltages*. Stevenage, England: Institution of Engineering and Technology, 1988.

[31] P. Alto and P. Kulkarni, *Insulator Defect Image Dataset - v1.2 , Kaggle*. California USA: Electric Power Research Institute, 2020. [Online]. Available: https://kaggle.com/competitions/insulator-defect-detection/data

[32] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 4th ed. Upper Saddle River, NJ: Pearson, 2017.

[33] C. Flyability SA, "Visual Inspections: A Complete Guide." [Online]. Available: https://www.flyability.com/visual-inspection

[34] c. Flyability SA, "NDT (Non-Destructive Testing): A Complete Guide." [Online]. Available: https://www.flyability.com/ndt

[35] "DroneQ robotics," https://www.droneq.nl/blog/?lang=en.

[36] H. Miura, A. Watanabe, M. Okugawa, T. Miura, Disaster Prevention Research Center, Aichi Institute of Technology 1247 Yachigusa, Yakusa-cho, Toyota-shi, Aichi 470-0392, Japan, Department of Mechanical Engineering, Faculty of Engineering, Aichi Institute of Technology 1247 Yachigusa, Yakusa-cho, Toyota-shi, Aichi 470-0392, Japan, and Department of Solution, Sanritz Automation Co., Ltd. 7-47-1 Kotobuki-cho, Toyota-shi, Aichi 471-0834, Japan, "Verification and evaluation of robotic inspection of the inside of culvert pipes," *J. Robot. Mechatron.*, vol. 31, no. 6, pp. 794–802, 2019.

[37] S. Anand and L. Priya, *A guide for machine vision in quality control*, 1st ed. Boca Raton: CRC Press, 2019.

[38] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. PP, pp. 1–1, 03 2020.

[39] "C-THRU: Ironman for Firefighters." [Online]. Available: https://www.youtube.com/watch?v=f1ZwVVoyk54

[40] Z. Long, X. Qiu, C. L. J. Chan, Z. Sun, Z. Yuan, S. Poddar, Y. Zhang, Y. Ding, L. Gu, Y. Zhou, W. Tang, A. K. Srivastava, C. Yu, X. Zou, G. Shen, and Z. Fan, "A neuromorphic bionic eye with filter-free color vision using hemispherical perovskite nanowire array retina," *Nature Communications*, vol. 14, no. 1, p. 1972, Apr. 2023. [Online]. Available: https://www.nature.com/articles/s41467-023-37581-y

[41] "Vision Is Our Dominant Sense | BrainLine," Nov. 2008. [Online]. Available: https://www.brainline.org/article/vision-our-dominant-sense

[42] K. A. Hussey, S. E. Hadyniak, and R. J. Johnston, "Patterning and Development of Photoreceptors in the Human Retina," *Frontiers in Cell and Developmental Biology*, vol. 10, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fcell.2022.878350

[43] S. Yantis and R. A. Abrams, *Sensation and perception*, second edition ed. New York, NY: Worth Publishers, Macmillan Learning, 2017, oCLC: ocn945745533.

[44] "VISION - imagia." [Online]. Available: https://afterimagia.pl/vision/

[45] M. Elgendy, *Deep learning for vision systems*. New York, NY: Manning Publications, 2021.

[46] M. L. Mastery, "14 different types of learning in machine learning," Blog. [Online]. Available: https://machinelearningmastery.com/types-of-learning-in-machine-learning/

[47] "What does this have to do with the brain? - Deep Neural Networks." [Online]. Available: https://www.coursera.org/lecture/neural-networks-deep-learning/what-does-this-have-to-do-with-the-brain-obJnR

[48] "Geogebra classic." [Online]. Available: https://www.geogebra.org/classic

[49] Wikipedia contributors, "CORDIC," https://en.wikipedia.org/w/index.php?title=CORDIC&oldid=1160255882, Jun. 2023, accessed: 2023-6-15.

[50] "3Blue1Brown - What is backpropagation really doing?" [Online]. Available: https://www.3blue1brown.com/lessons/backpropagation

[51] "Neural Style Transfer." [Online]. Available: https://cfml.se/blog/neural_style_transfer/

[52] "Stride Length and Dropout for Better Deep Learning Models | Kaggle." [Online]. Available: https://www.youtube.com/watch?v=fwNLf4t7MR8&list=PLqFaTIg4myu_4k32hrv_bCren2yB9xu4s&index=8

[53] H. Roth, A. Farag, L. Lu, B. Turkbey, and R. Summers, "Deep convolutional networks for pancreas segmentation in ct imaging," vol. 9413, 04 2015.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf, accessed: 2023-11-28.

[56] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[59] Z. Zou, K. Chen, Z. Shi, Member, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," accessed: 2023-12-5.

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 2016, arXiv:1506.01497 [cs]. [Online]. Available: http://arxiv.org/abs/1506.01497

[61] "Getting started with mask R-CNN for instance segmentation - MATLAB & simulink - MathWorks," https://ww2.mathworks.cn/help/vision/ug/getting-started-with-mask-r-cnn-for-instance-segmentation.html, accessed: 2023-12-5.

[62] S. Shin, H. Han, and S. H. Lee, "Improved YOLOv3 with duplex FPN for object detection based on deep learning," *The International Journal of Electrical Engineering & Education*, p. 002072092098352, Jan. 2021. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0020720920983524

[63] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," 2016.

[64] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," https://docs.ultralytics.com/models/yolov8/, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[65] INSAID, "Automated Hyperparameter tuning," Jan. 2022. [Online]. Available: https://insaid.medium.com/automated-hyperparameter-tuning-988b5aeb7f2a

[66] W. Koehrsen, "Overfitting vs. Underfitting: A Complete Example," Jan. 2018. [Online]. Available: https://towardsdatascience.com/overfitting-vs-underfitting-a-complete-example-d05dd7e19765

[67] "Underfitting vs. Overfitting." [Online]. Available: https://scikit-learn/stable/auto_examples/model_selection/plot_underfitting_overfitting.html

[68] I. Salehin and D.-K. Kang, "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain," *Electronics*, vol. 12, no. 14, p. 3106, Jan. 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/14/3106

[69] Shankar297, "Understanding Loss Function in Deep Learning," Jun. 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/

[70] A. Geron, *Hands on machine learning with ScikitLearn, Keras, and TensorFlow: concepts, tools, and techniques to build intelligent systems*, second edition ed. Beijing [China] ; Sebastopol, CA: O'Reilly Media, Inc, 2019.

[71] S. Narkhede, "Understanding AUC - ROC Curve," Jun. 2021. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[72] T. S. Teja, "Top Notebooks used by Machine Learning Engineers and Data Scientists." [Online]. Available: https://medium.com/@suryateja233/top-notebooks-used-by-machine-learning-engineers-and-data-scientists-ed025c02a2b6

[73] "Python (programming language)," Nov. 2023, page Version ID: 1185632218. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1185632218

[74] Y. Noema, "Top 3 Programming Languages For Implementing a CV System." [Online]. Available: https://medium.com/imagescv/my-top-3-programming-languages-for-implementing-a-computer-vision-system-2aa0a3ad0a2a

[75] "Project Jupyter," Nov. 2023, page Version ID: 1183623507. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Project_Jupyter&oldid=1183623507

[76] "tf.keras.utils.image_dataset_from_directory | TensorFlow v2.14.0." [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/utils/

image_dataset_from_directory

[77] "Python lambda/ function (with examples)," https://www.programiz.com/python-programming/anonymous-function, accessed: 2023-11-30.

[78] "Data augmentation," https://www.tensorflow.org/tutorials/images/data_augmentation, accessed: 2023-11-30.

[79] "The sequential model," https://www.tensorflow.org/guide/keras/sequential_model, accessed: 2023-11-30.

[80] "Tf.Keras.Layers.Conv2D," https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D, accessed: 2023-12-1.

[81] "Tf.Keras.Applications.Vgg16.VGG16," https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16, accessed: 2023-11-30.

[82] P. Gavrikov, "visualkeras," https://github.com/paulgavrikov/visualkeras, 2020.

[83] "Tf.Keras.Regularizers.Regularizer," https://www.tensorflow.org/api_docs/python/tf/keras/regularizers/Regularizer, accessed: 2023-12-1.

[84] "Tf.Keras.Callbacks.ModelCheckpoint," https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint, accessed: 2023-12-1.

[85] "Tf.Keras.Model," https://www.tensorflow.org/api_docs/python/tf/keras/Model, accessed: 2023-12-1.

[86] "Pyplot tutorial — matplotlib 3.8.2 documentation," https://matplotlib.org/stable/tutorials/pyplot.html, accessed: 2023-12-1.

[87] "glob — unix style pathname pattern expansion," https://docs.python.org/3/library/glob.html, accessed: 2023-12-1.

[88] "Classification_report_imbalanced — version 0.11.0," https://imbalanced-learn.org/stable/references/generated/imblearn.metrics.classification_report_imbalanced.html, accessed: 2023-12-1.

[89] "Multiclass receiver operating characteristic (ROC)," https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html, accessed: 2023-12-1.

[90] "Numpy.Arange — NumPy v1.26 manual," https://numpy.org/doc/stable/reference/generated/numpy.arange.html, accessed: 2023-12-1.

[91] "Confusion matrix," https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html, accessed: 2023-12-1.

[92] "Seaborn.Heatmap — seaborn 0.13.0 documentation," https://seaborn.pydata.org/generated/seaborn.heatmap.html, accessed: 2023-12-1.

[93] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," http://arxiv.org/abs/1610.02391, accessed: 2023-12-1.

[94] "Image data explanation benchmarking: Image multiclass classification — SHAP latest documentation," https://shap.readthedocs.io/en/latest/example_notebooks/benchmarks/image/Image%20Multiclass%20Classification%20Benchmark%20Demo.html, accessed: 2023-12-1.

[95] "LIME for image classification — OmniXAI documentation," https://opensource.salesforce.com/OmniXAI/latest/tutorials/vision/lime.html, accessed: 2023-12-1.

[96] V. Kamakshi and N. C. Krishnan, "Explainable Image Classification: The Journey So Far and the Road Ahead," *AI*, vol. 4, no. 3, pp. 620–651, Sep. 2023. [Online]. Available: https://www.mdpi.com/2673-2688/4/3/33

[97] "Grad-CAM class activation visualization," https://keras.io/examples/vision/grad_cam/, accessed: 2023-12-1.

[98] "A dataset of functional and defective solar cells extracted from EL images of solar modules," https://github.com/zae-bayern/elpv-dataset/blob/master/README.md.

[99] M. Millendorf, E. Obropta, and N. Vadhavkar, "Infrared solar module dataset for anomaly detection," 2020.

[100] D. Lewis and P. Kulkarni, "Insulator defect detection," Electric Power Research Institute, IEEE Dataport, 2021. doi:10.21227/vkdw-x769. [Online]. Available: https://ieee-dataport.org/competitions/insulator-defect-detection

[101] S. Deitsch, V. Christlein, S. Berger, C. Buerhop-Lutz, A. Maier, F. Gallwitz, and C. Riess, "Automatic classification of defective photovoltaic module cells in electroluminescence images," *Solar Energy*, vol. 185, pp. 455–468, Jun. 2019.

[102] S. Deitsch, C. Buerhop-Lutz, E. Sovetkin, A. Steland, A. Maier, F. Gallwitz, and C. Riess, "Segmentation of photovoltaic module cells in uncalibrated electroluminescence images," vol. 32, no. 4.

[103] H. Shin, J. Kang, and O. Gungor, "Automatic classification of defective photovoltaic module cells in electroluminescence images," http://noiselab.ucsd.edu/ECE228-2020/projects/Report/41Report.pdf, accessed: 2023-12-12.

[104] L. Pratt, J. Mattheus, and R. Klein, "A benchmark dataset for defect detection and classification in electroluminescence images of PV modules using semantic segmentation," *Systems and Soft Computing*, vol. 5, no. 200048, p. 200048, 2023.

[105] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," http://arxiv.org/abs/1708.02002, accessed: 2024-1-16.

[106] V. Vito and L. Y. Stefanus, "An asymmetric contrastive loss for handling imbalanced datasets," http://arxiv.org/abs/2207.07080, accessed: 2024-1-16.

[107] "Over-sampling methods — version 0.11.0," https://imbalanced-learn.org/stable/references/over_sampling.html, accessed: 2024-1-16.

[108] A. Chattopadhyay, A. Sarkar, Member, P. Howlader, V. N. Balasubramanian, and Member, "Grad-CAM++: Improved visual explanations for deep convolutional networks," http://arxiv.org/abs/1710.11063, accessed: 2024-1-28.

[109] SamsonWoof, "Tf_keras_gradcamplusplus: Tensorflow.Keras implementation of gradcam and gradcam++."

[110] Y. Zefri, I. Sebari, H. Hajji, and G. Aniba, "Developing a deep learning-based layer-3 solution for thermal infrared large-scale photovoltaic module inspection from orthorectified big UAV imagery data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 106, p. 102652, Feb. 2022.

[111] H. Cheng, Y. Zhai, R. Chen, D. Wang, Z. Dong, and Y. Wang, "Self-Shattering Defect Detection of Glass Insulators Based on Spatial Features," *Energies*, vol. 12, no. 3, p. 543, Feb. 2019. [Online]. Available: http://www.mdpi.com/1996-1073/12/3/543

[112] R. M. Prates, R. Cruz, A. P. Marotta, R. P. Ramos, E. F. Simas Filho, and J. S. Cardoso, "Insulator visual non-conformity detection in overhead power distribution lines using deep learning," *Computers & Electrical Engineering*, vol. 78, pp. 343–355, Sep. 2019. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S004579061930967X

[113] Z. Dai, "Uncertainty-aware accurate insulator fault detection based on an improved YOLOX model," *Energy Reports*, vol. 8, pp. 12 809–12 821, Nov. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2352484722019205

[114] M. Hanoi, "Faster r-cnn using detectron2," Vietnam, 2023. [Online]. Available: https://www.kaggle.com/code/monkira/faster-rcnn

[115] "Edplabelec." [Online]. Available: https://www.edp.com/pt-pt/inovacao/labelec/testes-e-ensaios

[116] Ultralytics, "Augmentation configuration." [Online]. Available: https://docs.ultralytics.com/usage/cfg/#augmentation

[117] R. Levy, "YOLOv8 Finetuning for PPE detection," Rio de Janeiro. [Online]. Available: https://kaggle.com/code/hinepo/yolov8-finetuning-for-ppe-detection

[118] R. King, "Brief summary of yolov8 model structure - issue #189," https://github.com/ultralytics/ultralytics/issues/189, 2023.

[119] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," 2019. [Online]. Available: https://github.com/facebookresearch/detectron2

[120] W. Feng, X. Zhang, and G. Zhao, "ResNetX: a more disordered and deeper network architecture," Dec. 2019, arXiv:1912.12165 [cs, eess]. [Online]. Available: http://arxiv.org/abs/1912.12165

[121] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," Apr. 2017, arXiv:1611.05431 [cs]. [Online]. Available: http://arxiv.org/abs/1611.05431

[122] L. Das, M. H. Saadat, B. Gjorgiev, E. Auger, and G. Sansavini, "Object detection-based inspection of power line insulators: Incipient fault detection in the low data-regime," Dec. 2022, arXiv:2212.11017 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2212.11017

[123] T. Wang, Y. Zhai, Y. Li, W. Wang, G. Ye, and S. Jin, "Insulator Defect Detection Based on ML-YOLOv5 Algorithm," *Sensors (Basel, Switzerland)*, vol. 24, no. 1, p. 204, Dec. 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10781232/

Appendix

Appendix A - Photo-Voltaic Cells EL Images

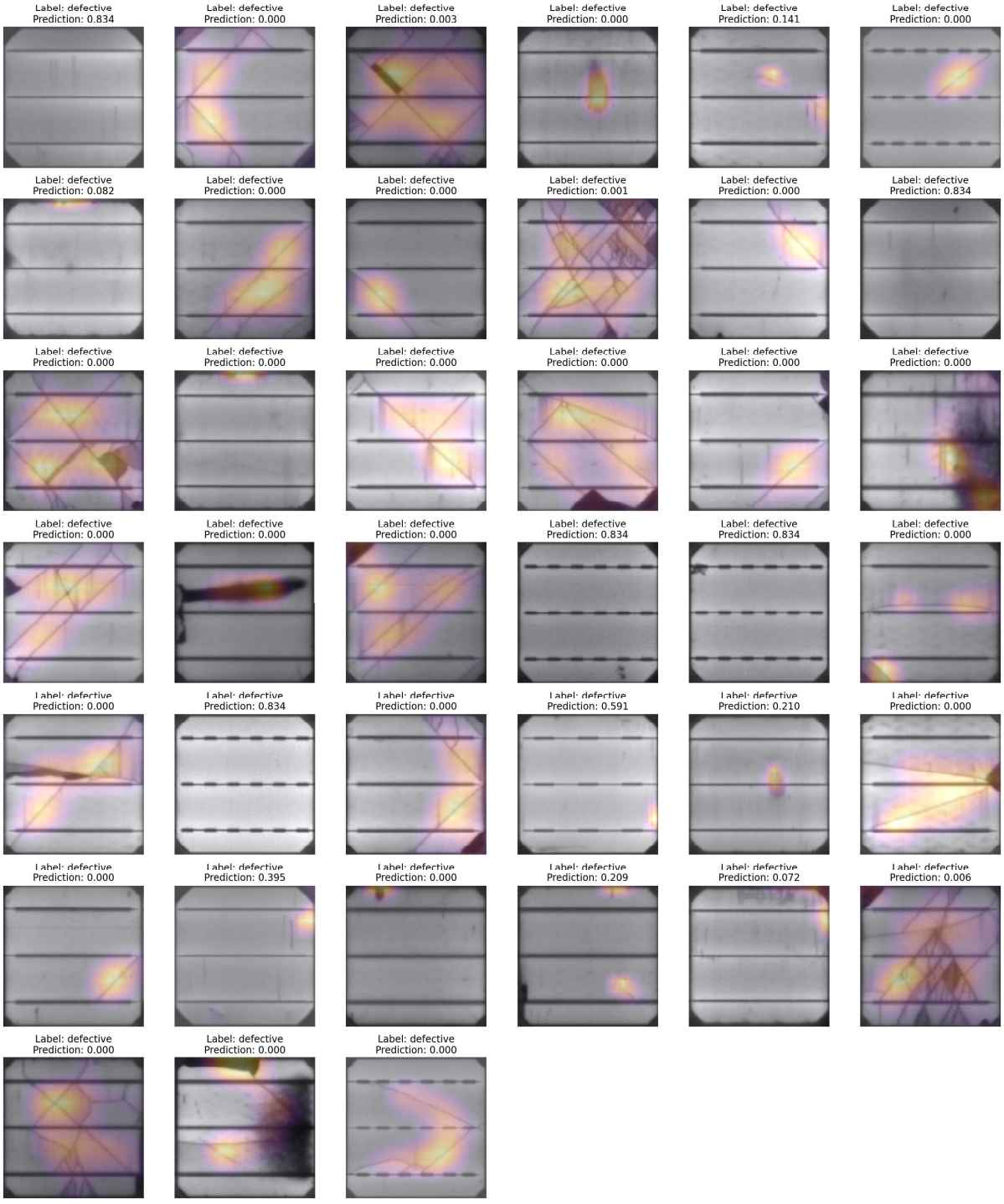


Figure 5.1: Defective Monocrystalline PV Cells test inference results and heatmap

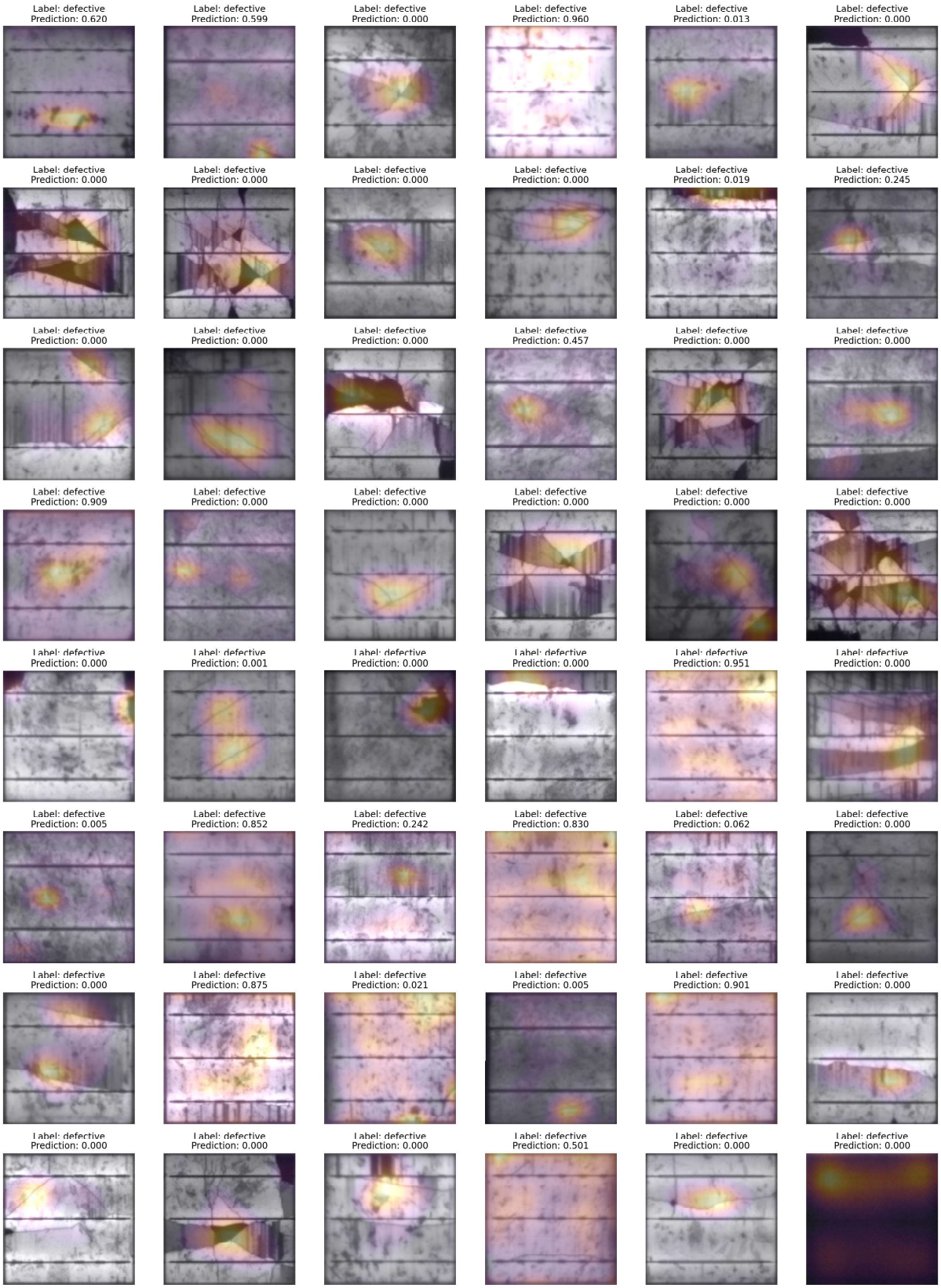


Figure 5.2: Defective Polycrystalline PV Cells test inference results and heatmap

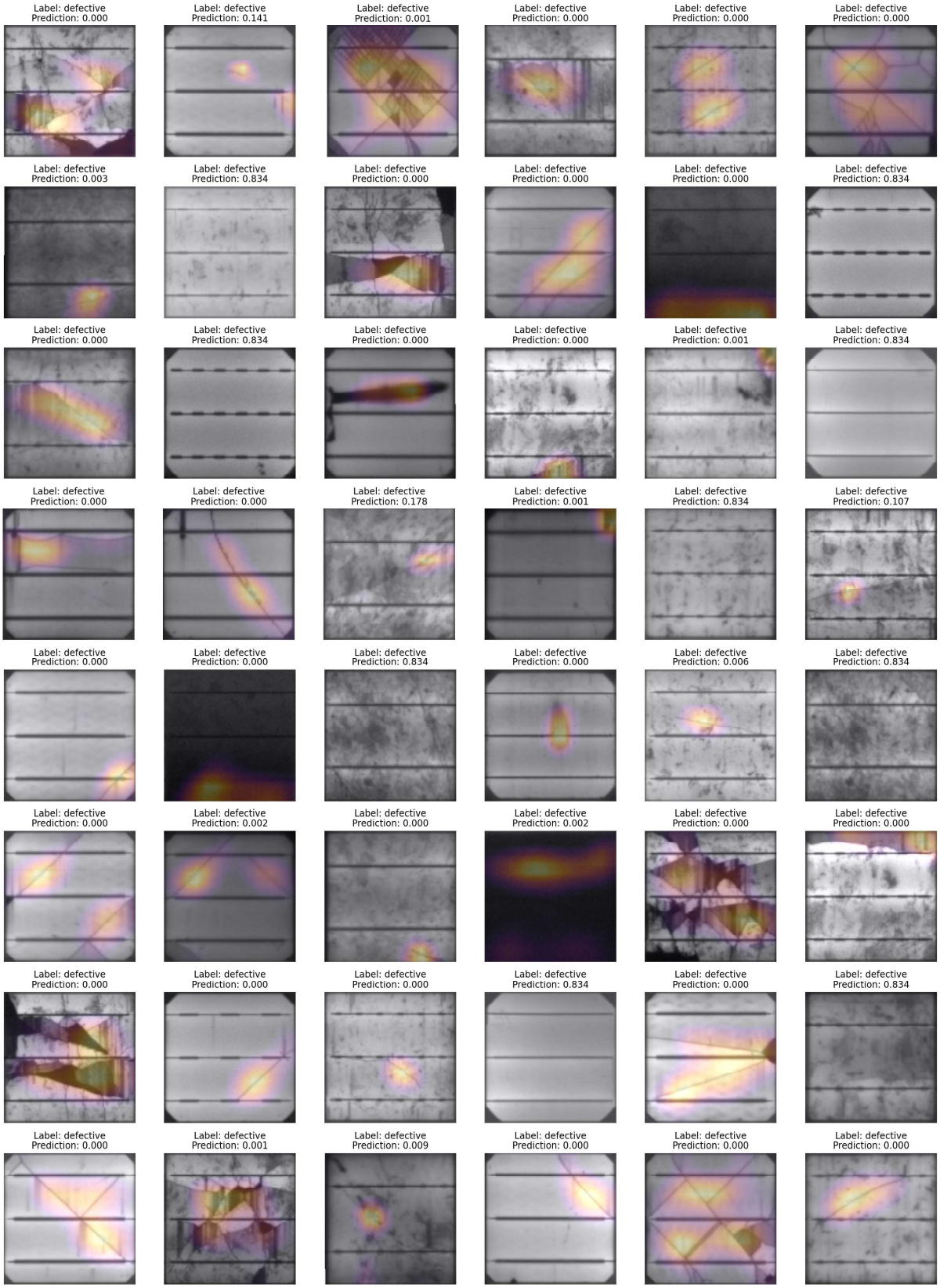


Figure 5.3: Defective PV Cells test inference results and heatmap

Appendix B - Photo-Voltaic Modules IR Images

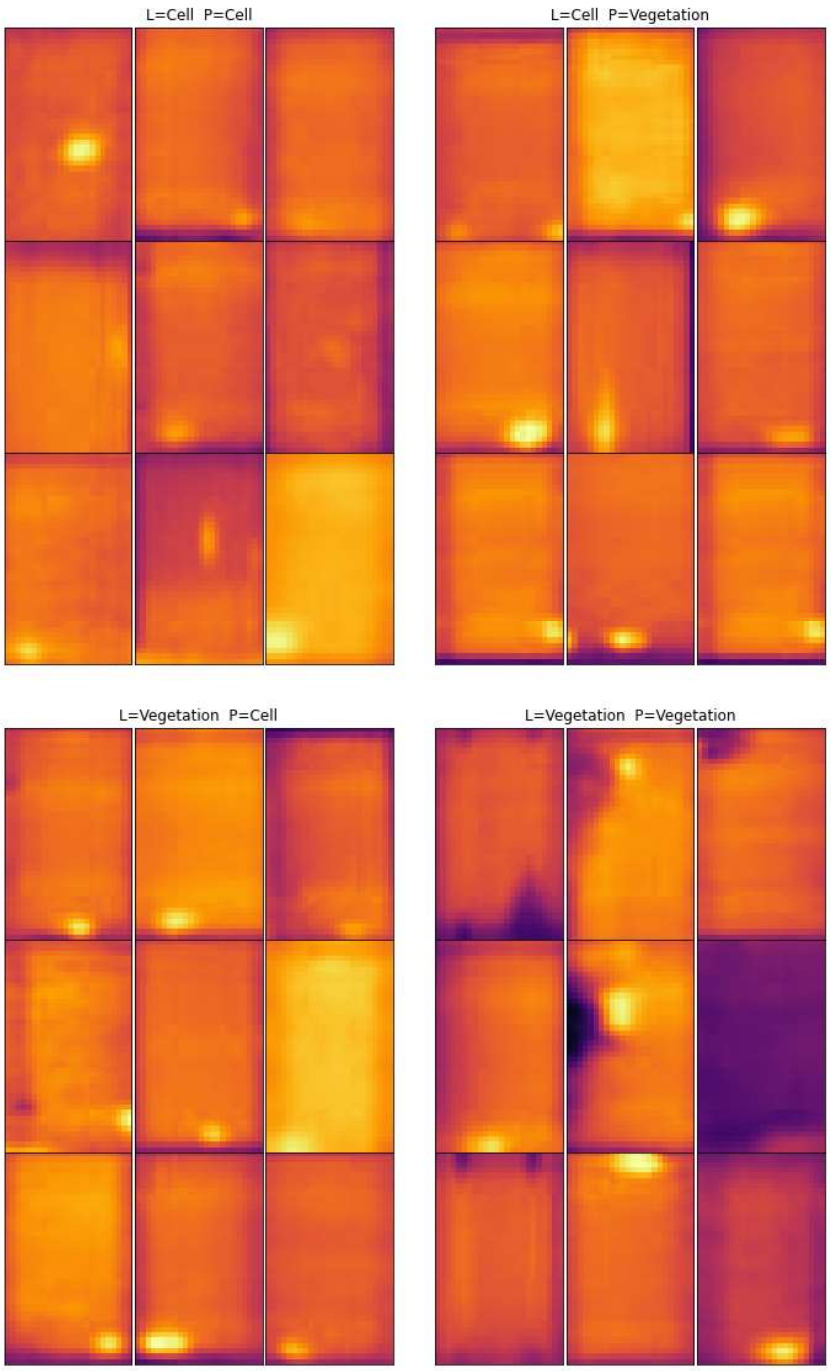


Figure 5.4: Confusion matrix with images from classes Cell and Vegetation

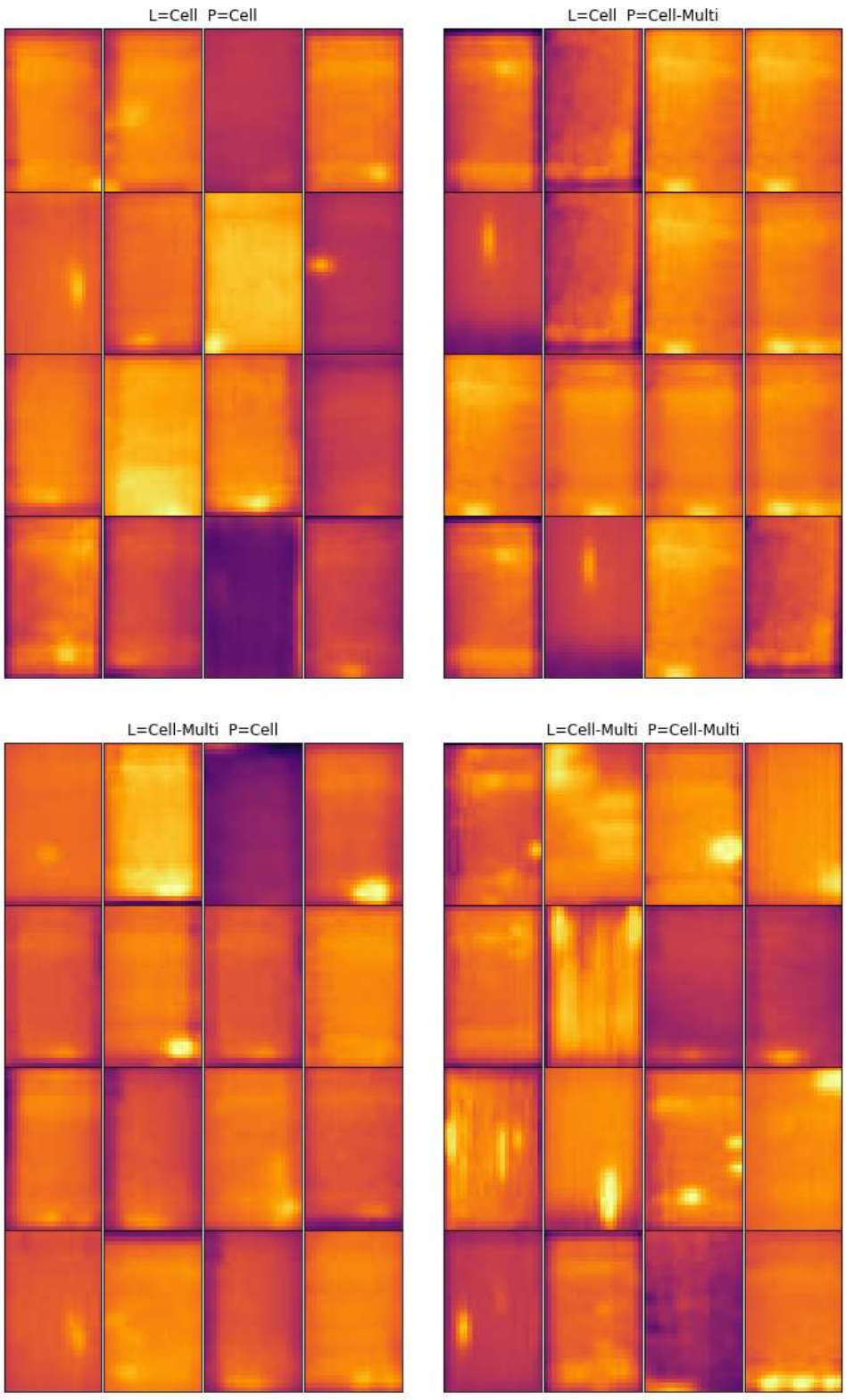


Figure 5.5: Confusion matrix with images from classes Cell and Cell-Multi

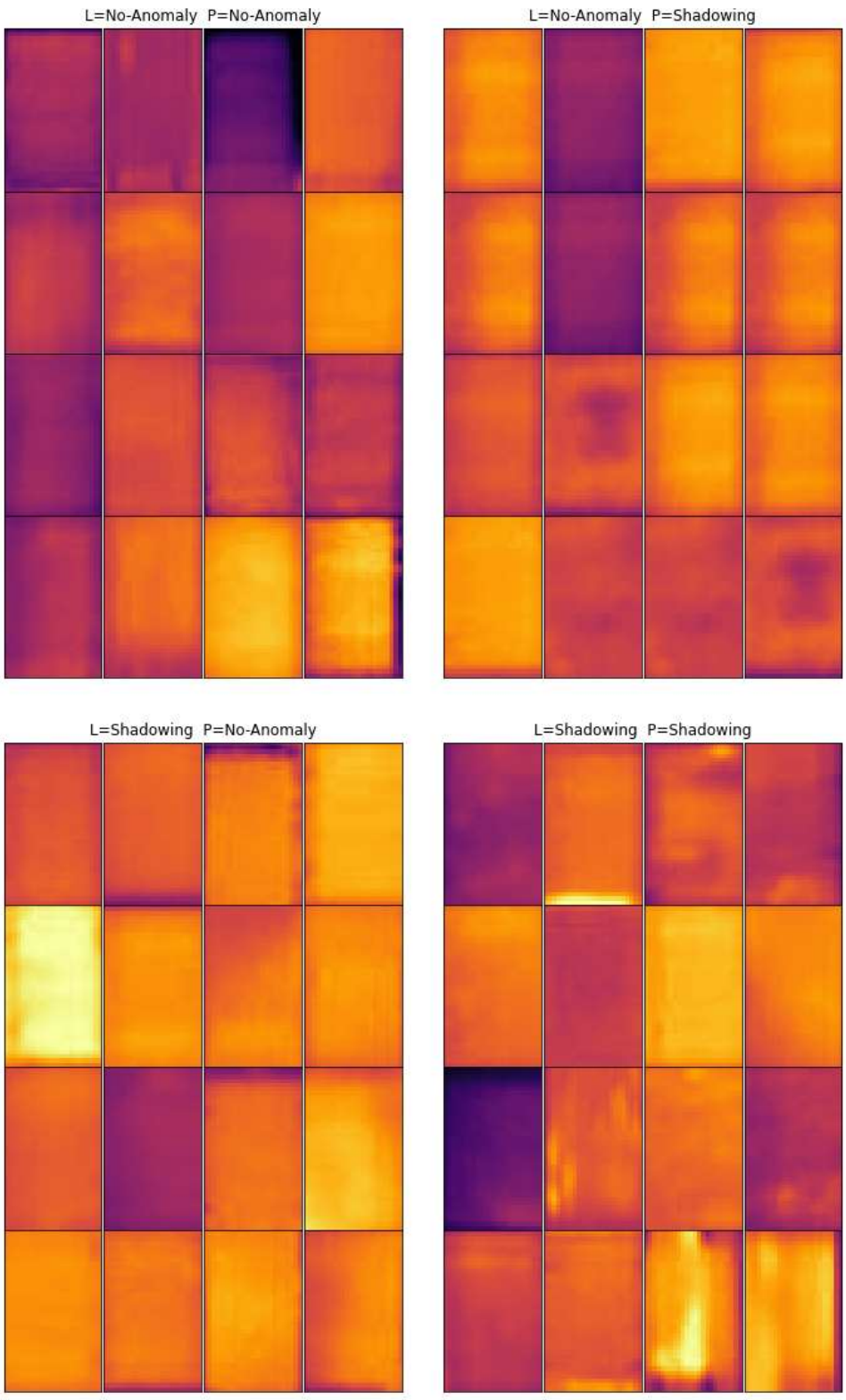


Figure 5.6: Confusion matrix with images from classes No-Anomaly and Shadowing

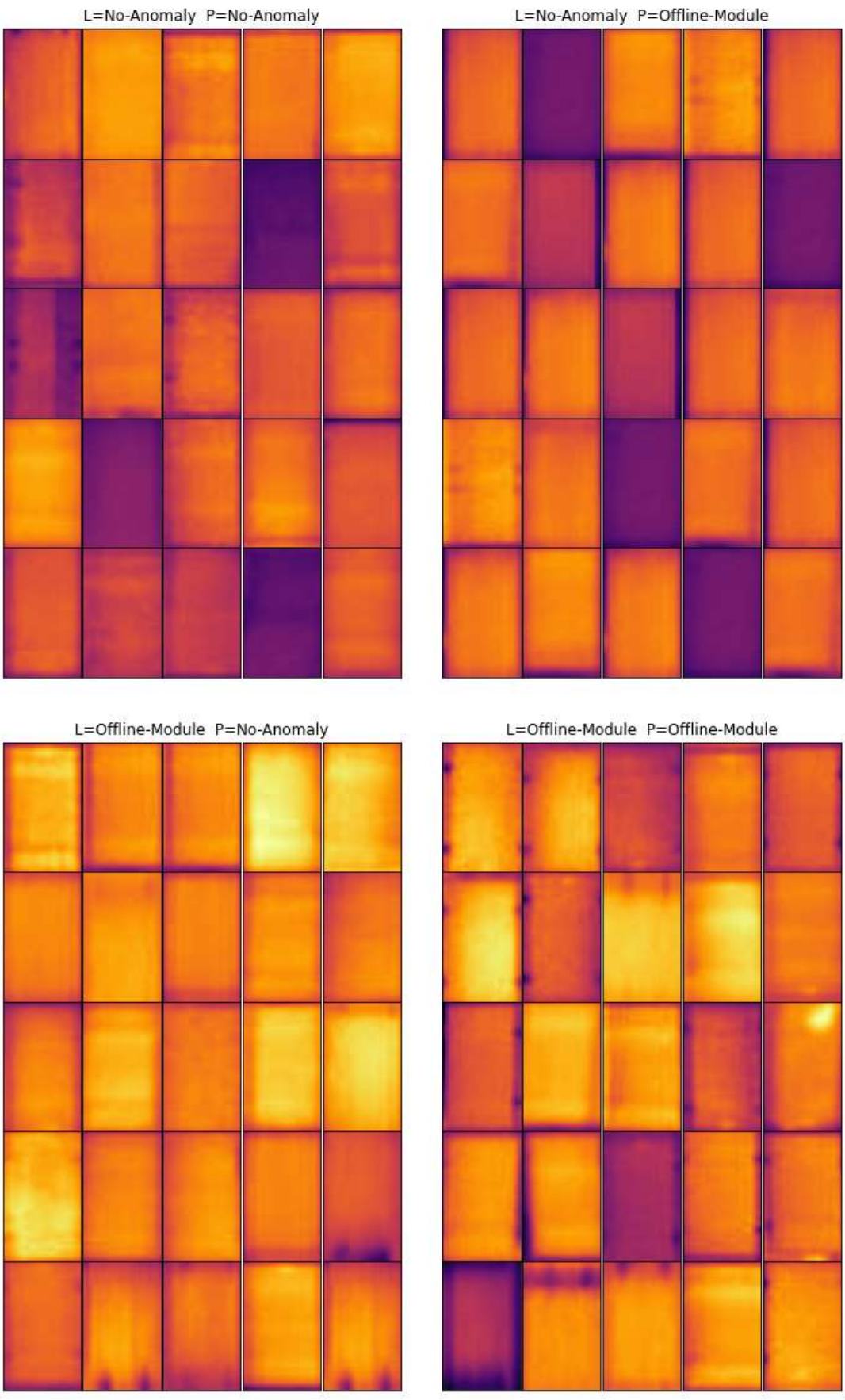


Figure 5.7: Confusion matrix with images from classes No-Anomaly and Offline-Module

Appendix C - High Voltage Insulators RGB Images

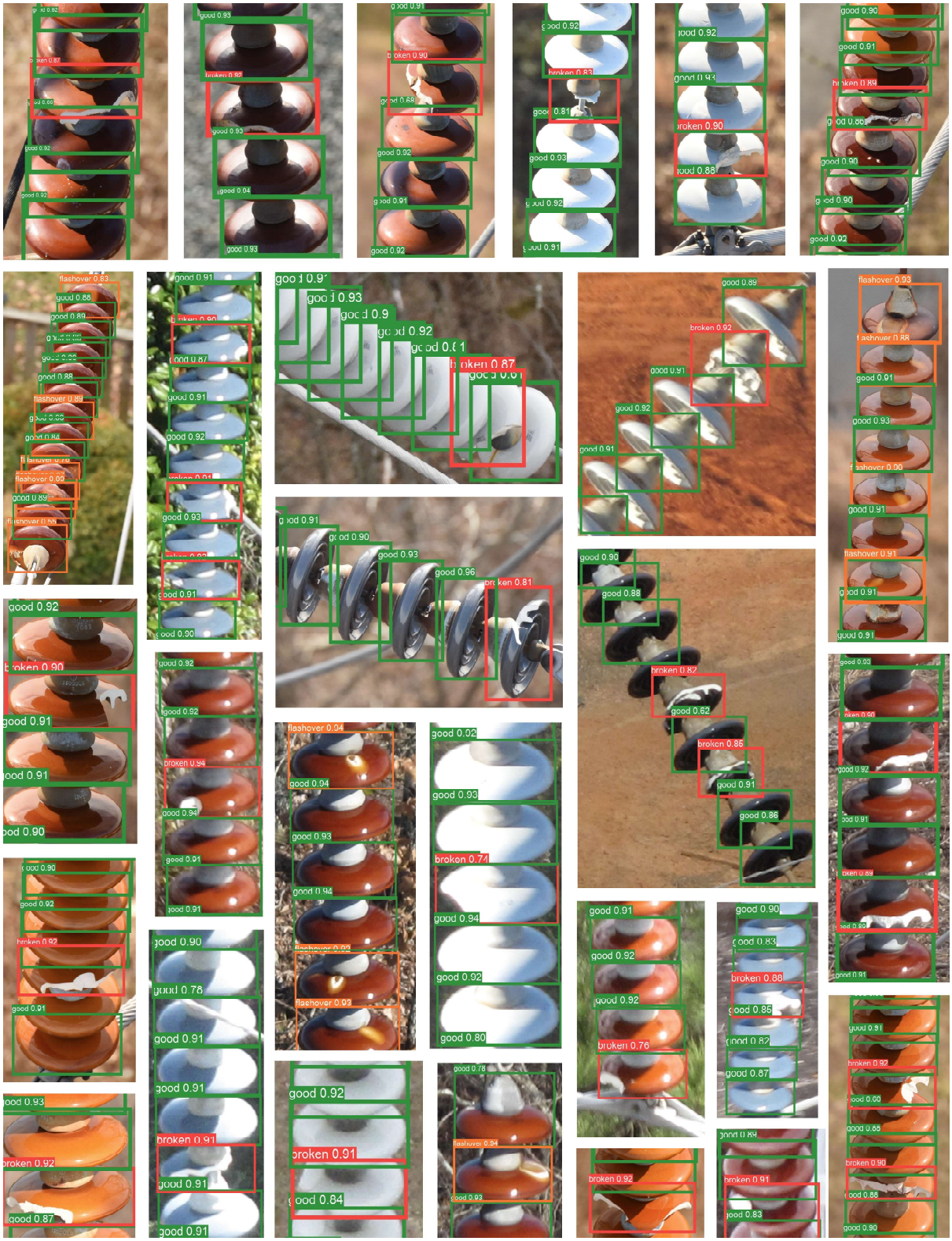


Figure 5.8: Inference test IDID using YOLO

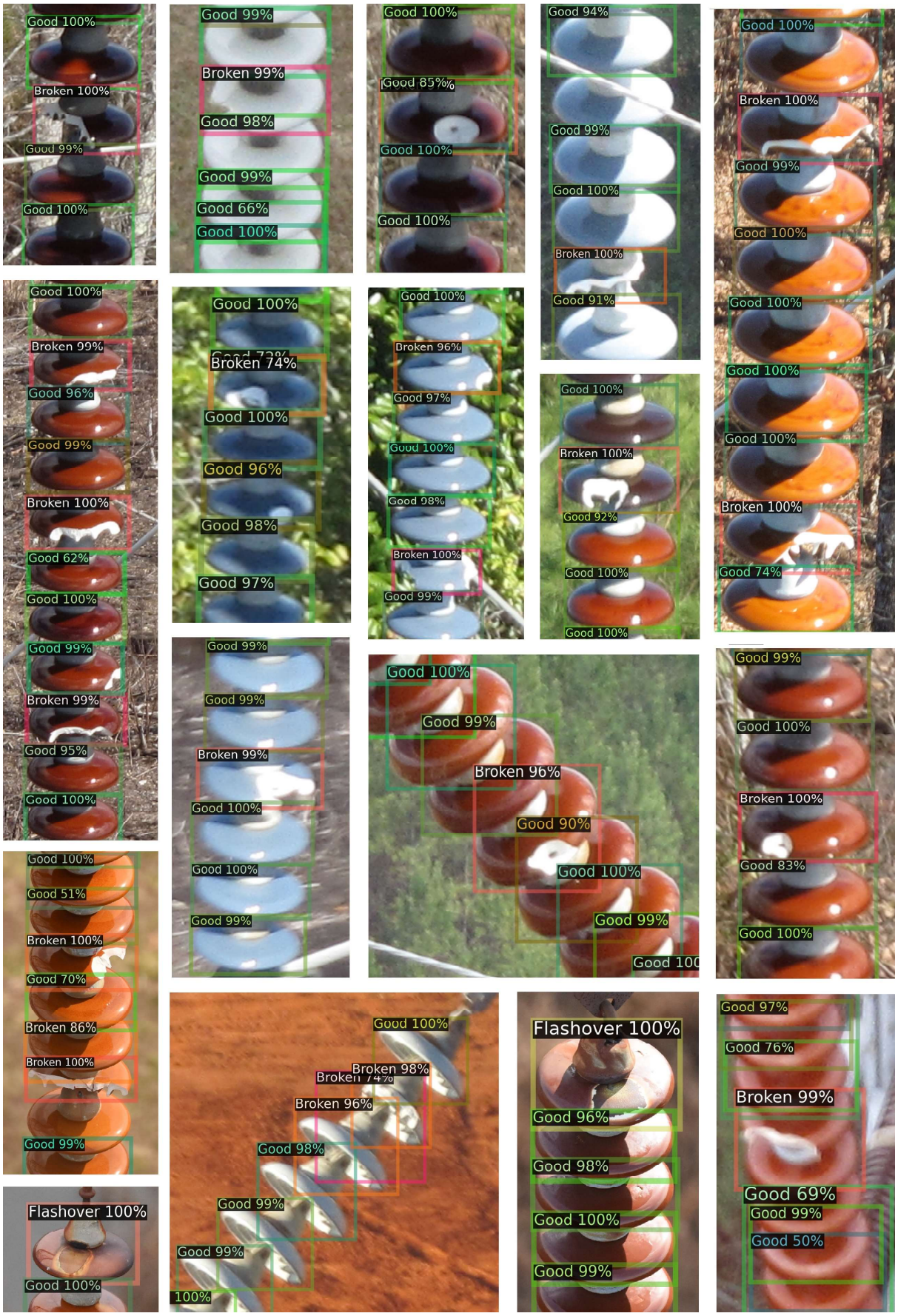


Figure 5.9: Inference test IDID using Faster R-CNN

Instituto Superior
de Engenharia
Politécnico de Coimbra