Technical University of Crete

School of Electrical and Computer engineering

DIPLOMA THESIS

PHOTOVOILTAIC PANEL INSPECTION USING DEEP LEARNING SYSTEMS ON UNMANNED AERIAL VEHICLE BASED IMAGES

Dimopoulos Konstantinos

Committee

Professor Michail Zervakis (Supervisor)

Professor Fotios Kanellos

Professor Panagiotis Partsinevelos

Chania, May 2024

Πολυτεχνείο Κρήτης

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΘΕΩΡΗΣΗ ΦΩΤΟΒΟΛΤΑΙΚΩΝ ΣΤΟΙΧΕΙΩΝ ΜΕ ΤΗΝ ΒΟΗΘΕΙΑ ΣΥΣΤΗΜΑΤΩΝ ΒΑΘΕΙΑΣ ΜΑΘΗΣΗΣ ΣΕ ΕΙΚΟΝΕΣ ΑΠΟ ΜΗ ΕΠΑΝΔΡΟΜΕΝΟ ΟΧΗΜΑ

Δημόπουλος Κωνσταντίνος

Επιτροπή

Καθηγητής Μιχάλης Ζερβάκης (Επιβλέπων)

Καθηγητής Φώτιος Κανέλλος

Καθηγητής Παναγιώτης Παρτσινέβελος (Τμήμα ΜΗΧΟΠ)

Χανία, Μάιος 2024

Acknowledgments

I would like to thank Professor Michail Zervakis and Dr. Marios Antonakakis for the guidance and assistance that they provided to me throughout the whole experimentation process.

I would also like to thank the members of the committee, Professor Fotios Kanellos and Professor Panagiotis Partsinevelos

Finally, special thanks should go to my family, whose unwavering support and constant encouragement fueled my perseverance during both the completion of this project and all the years of my tuition in this institution.

Abstract

It is apparent to everyone that the world that we live in is slowly but steadily transitioning from fossil fuels to renewable and clean energy. This is quite beneficial for the environment, the humans that inhabit this planet and the fauna and flora as well. One of the most rapidly growing sources of green energy is solar anergy. Constructions of huge solar farms are taking place each year. With each farm containing hundreds of thousands of individual photovoltaic panels (PV), the need for a faster, safer, and more efficient way of condition monitoring for each panel has risen. Surveillance monitoring via the use of unmanned aerial vehicles (UAVs) has been proposed and applied, showing exceptional results. In this thesis, the focus is on investigating the use of novel deep learning models, that take as an input infrared spectrum images of individual photovoltaic panels with non-annotated multiple damages. The goal is first to create an annotated dataset for training, validation and testing purposed and a general data augmented dataset for further testing purposes. The second goal is to achieve an optimal training of 2 individually trained models. The first of the models excels in binary detection of surface of solar panels, meaning that it can detect efficiently whether a photovoltaic panel is damaged or not. The other model excels at multi-class classification, meaning that it can detect, identify and classify the specific error on a damaged panel, and then proceeds to put a bounding box over the error, pinpointing its exact location on the image. Throughout this this sensitivity investigation, the YOLOv8 and YOLONAS models are used. For YOLOv8, the detection performance on the training phase achieved for the binary model an F1 score of 93.8% and mAP50-90 score of 97.7%. Our multi-class classification model can correctly identify 8 distinct categories of solar panel surface faults, with different percentages of success depending on the fault class in question (e.g. F1 score: 64.4% and mAP50-90 score: 45.3%). The specs of these trained models are further analyzed in this thesis, together with all the process that was required to reach this point. These models were tested both on part of the initial database as well as a smaller database that was created using data augmentation. For first time, two cutting-edge DNN models were used for multiclass detection of damages in PV panels paving the way for future more efficient and massive detection of damages in PV farms.

Περίληψη

Είναι προφανές σε όλους ότι ο κόσμος στον οποίο ζούμε μεταβαίνει αργά αλλά σταθερά από τα ορυκτά καύσιμα στις ανανεώσιμες πηγές ενέργεια. Αυτό είναι αρκετά ωφέλιμο για το περιβάλλον, τους ανθρώπους που κατοικούν σε αυτόν τον πλανήτη, καθώς και την πανίδα και τη χλωρίδα . Μία από τις πιο ταχέως αναπτυσσόμενες πηγές πράσινης ενέργειας είναι η ηλιακή ενέργεια. Κατασκευές τεράστιων ηλιακών πάρκων πραγματοποιούνται κάθε χρόνο. Καθώς κάθε ηλιακό πάρκο περιέχει εκατοντάδες χιλιάδες μεμονωμένα ηλιακά πάνελ, έχει αυξηθεί η ανάγκη για έναν ταχύτερο, ασφαλέστερο και πιο αποτελεσματικό τρόπο παρακολούθησης της κατάστασης του κάθε πάνελ. Έχει προταθεί και εφαρμοστεί η εξ αποστάσεως παρακολούθηση των πάνελ μέσω της χρήσης μη επανδρωμένων εναέριων οχημάτων (UAV ή αλλιώς drones), παράγοντας εξαιρετικά αποτελέσματα. Σε αυτή τη διατριβή εστιάζουμε στη δημιουργία καινοτόμων μοντέλων βαθιάς εκμάθησης, που λαμβάνουν ως είσοδο, εικόνες μεμονωμένων φωτοβολταϊκών πάνελ που λαμβάνονται, των οποίων οι βλάβες δεν έχουν σχολιασθεί. Οι εικόνες αυτές είναι εικόνες του υπέρυθρου φάσματος του φωτός. Έχουμε ως στόχο την δημιουργία ενός σχολιασμένου συνόλου δεδομένων, για την εκπαίδευση, την επικύρωση και τον έλεγχο των μοντέλων καθώς και η δημιουργία ενός γενικού συνόλου δεδομένων, μέσω data augmentation, για χρήση σε περεταίρω ελέγχους. Δεύτερο στόχο αποτελεί η επίτευξη της αποδοτικής εκπαίδευσης 2 διαφορετικών μοντέλων. Το πρώτο μοντέλο εκτελεί δυαδική ανίχνευση σφαλμάτων στην επιφάνειας των ηλιακών πάνελ, που σημαίνει ότι μπορεί να ανιχνεύσει αποτελεσματικά εάν ένα φωτοβολταϊκό πάνελ έχει υποστεί κάποια φθορά ή όχι. Το δεύτερο μοντέλο εξειδικεύεται στην ταξινόμηση πολλαπλών κατηγοριών σφαλμάτων, που σημαίνει ότι μπορεί να ανιχνεύσει, να αναγνωρίσει και να ταξινομήσει το συγκεκριμένο σφάλμα σε ένα πάνελ και, στη συνέχεια, προχωρά στην τοποθέτηση ενός πλαισίου οριοθέτησης πάνω από το σφάλμα, προσδιορίζοντας την ακριβή θέση του στην εικόνα. Σε όλο αυτό το πείραμα, τροποποιούμε τα μοντέλα YOLOv8 και YOLONAS. Κατά την εκπαίδευση του YOLOv8 εκπαιδεύσαμε ένα μοντέλο δυαδικής αναγνώρισης με F1 score ίσο με 93.8% και mAP50-90 score ίσο με 97.7%. Το μοντέλο ταξινόμησης πολλαπλών κατηγοριών, μπορεί να αναγνωρίσει σωστά 8 διαφορετικές κατηγορίες σφαλμάτων επιφάνειας ηλιακών πάνελ, με διαφορετικά ποσοστά επιτυχίας ανάλογα με την εν λόγω κατηγορία σφάλματος (με F1 score ίσο με 64.4% και mAP50-90 score ίσο με 45.3%). Οι προδιαγραφές αυτών των εκπαιδευμένων

μοντέλων αναλύονται περαιτέρω σε αυτή τη διατριβή, μαζί με όλη τη διαδικασία και τον πειραματισμό που χρειάστηκε για να φτάσουμε σε αυτό το σημείο. Τα μοντέλα ελέγχθηκαν με την χρήση τμήματος των αρχικών εικόνων αλλά και με εντελώς νέες εικόνες που δημιουργήθηκαν μέσω της τεχνικής data augmentation. Για πρώτη φορά 2 καινοτόμα μοντέλα βαθιάς μάθησης χρησιμοποιήθηκαν για τον εντοπισμό πολλαπλών κλάσεων σφαλμάτων σε φωτοβολταϊκά πάνελ, ανοίγοντας έτσι τον δρόμο για περεταίρω αποδοτική ανίχνευση σφαλμάτων σε μεγάλα ηλιακά πάρκα.

Table of Contents

CHAPTER 1 INTRODUCTION 22
1.1 UNMANNED AERIAL VEHICLES (UAVs) 22
1.2 INFRARED THERMOGRAPHY vs ELELECTOLUMINESCENT IMAGERY 23
1.3 ARIEAL INFRARED THERMOGRAPHY (AIRT) 24
1.4 CONVOLUTIONAL NEURAL NETWORKS (CNNs) 25
1.5 PAST WORK ON THE FIELD 26
1.6 NOVELTY OF OUR THESIS 28
CHAPTER 2 THEORY 29
2.1 Artificial Intelligence (AI) 29
2.2 MACHINE LEARNING 29
2.3 DEEP LEARNING 30
2.4 Convolutional Neural Network (CNN) 31
2.4.1 CONVOLUTION LAYER 32
2.4.2 POOLING LAYER 32
2.4.3 FULLY CONNECTED LAYER 34
2.5 ACTIVATION 34
2.5.1 BINARY STEP 35
2.5.2 LINEAR 36
2.5.3 SIGMOID 36
2.5.4 TANH 37
2.5.5 RELU 38
2.5.6 LEAKY RELU 39
2.5.7 PARAMETRIZED RELU FUNCTION 40
2.5.8 ELU 40
2.5.9 SWISH 41
2.5.10 SOFTMAX 42

2.6 LOSS FUNCTIONS 43
2.7 OPTIMIZER 44
2.8 HYPERPARAMETERS 45
2.8.1 LEARNING RATE 45
2.8.2 NUMBER OF EPOCHS 45
2.8.3 BATCH SIZE 46
2.9 ANALYSIS OF THE YOLO MODELS 46
2.9.1 APPLICATIONS 46
2.9.2 FIRST APPROACH (YOLOv1) 48
2.9.3 YOLOv8 49
2.9.4 YOLONAS 52
Chapter 3 DATASET 53
3.1 CELL 54
3.2 DIODE 55
3.3 HOT SPOTS 56
3.4 SHADOWING 56
3.5 OFFLINE MODULE 57
3.6 SOILING 58
Chapter 4 METRICS 59
4.1 THE CONFUSION MATRIX 59
4.2 ACCURACY 60
4.3 PRECISION 61
4.4 RECALL 61
4.5 F1 SCORE 62
4.6 INTERSECTION OVER UNION 63
4.7 MEAN AVERAGE PERCISION, mAP50 & mAP50-95 63
Chapter 5 EXPERIMENTATION PROCESS 64
5.1 HARDWARE THAT WAS USED 64
5.2 SORTING PROCESS 64

5.3 ANNOTATION PROCESS 65
5.4 FIRST TRAINING 66
5.5 INCLUSIVE DATASET 67
5.6 COMPLETE DATASET 68
5.7 YOLONAS MODEL 69
5.8 BINARY DETECTION MODEL 69
5.9 TESTING PROCESS 70
Chapter 6 RESULTS AND COMPARISONS 72
6.1 EPOCH COUNT ON THE SMALL DATASET 72
6.2 FURTHER ANOMALY CLASSES INCLUSION 74
6.3 POST DATASET MODIFICATION 75
6.4 MULTIPLE HYPERPARAMETER COMPARISON ON THE COMPLETE DATASET 76
6.4.1 OPTIMIZER COMPARISON 76
6.4.2 BATCH SIZE COMPARISON 77
6.4.3 LEARNING RATE COMPARISON 78
6.4.4 FINAL COMPARISONS FOR THE MULTI- CLASS CLASSIFICATION MODEL 78
6.5 YOLONAS COMPARISON 80
6.6 BINARY COMPARISON 82
6.7 COMPARING TEST RESULTS OF OUR MOST OPTIMIZED MODELS 88
6.7.1 COMPARING TEST RESULTS OF OUR MULTI-CLASS CLASSIFICATION MODEL 88
6.7.2 COMPARING TEST RESULTS OF OUR BINARY DETECTION MODEL 90
Chapter 7 DISCUSSION 91
7.1 GENERAL INTERPRETATIONS 91
7.2 COMPARISON TO PAST WORKS 92
7.2.1 COMPARISON WITH A MODEL TRAINED ON THE SAME DATASET 92
7.2.2 COMPARISON WITH A YOLOv8 TRAINED ON A DIFFERENT DATASET 93
7.3 LIMITATIONS 95
7.4 OUTLOOK AND POTENTIAL FUTURE WORK 95
7.5 CONCLUSION 96

Chapter 8 REFERENCES 97

Chapter 9 APPENDIX 106

9.1 INTITIAL TRAINING OF THE YOLOv8 MODEL ON A SMALLER DATABASE 106

9.2 TRAINING OF THE YOLOv8 MODEL ON A MORE INCLUSIVE DATASET 109

9.3 DATASET MODIFICATION AND RETRAINING OF THE YOLOv8 MODEL 110

9.4 TRAINING OF THE YOLOv8 MODEL ON THE COMPLETE DATASET 112

9.5 TRAINING OF THE YOLONAS MODEL ON THE COMPLETE DATASET 124

9.6 TRAINING OF THE YOLOv8 MODEL FOR BINARY DETECTION 125

9.7 TESTING THE MOST OPTIMIZED MODELS 132

Table of Figures

Figure 1 Image of a UAV performing AIRT 25
Figure 3 Visualization of the differences between AI, ML and DL 30
Figure 4 Simplified visualization of a convolutional neural network structure 31
Figure 5 Visualization of the 2 different pooling processes 33
Figure 6 The process that a CNN goes through in the convolutional and pooling layers 33
Figure 7 Plot of the binary step activation function 35
Figure 8 Plot of the linear activation function 36
Figure 9 Plot of the sigmoid activation function 37
Figure 10 Plot of the tanh activation function 38
Figure 11 Plot of the relu activation function 39
Figure 12 Plot of the parametrized relu activation function 40
Figure 13 Plot of the ELU activation function 41
Figure 14 Plot of the swiss activation function 41
Figure 15 Plot of the Softmax activation function 42
Figure 16 Bibliometric network visualization of the main YOLO Applications 48
Figure 17 YOLO output prediction. The figure depicts a simplified YOLO model with a three-by-three grid, three classes, and a single class prediction per grid element to produce a vector of eight values. 49
Figure 18 Image representation of the YOLOv8 model architecture 50
Figure 19 Details of the previously represented YOLOv8 model architecture 51
Figure 20 Performance comparison of YOLO object detection models. The left plot illustrates the relationship between model complexity (measured by the number of parameters) and detection accuracy (COCO mAP50-95). The right plot shows the tradeoff between inference speed and accuracy for the same models 51
Figure 21 Composition of the images in our training dataset 53
Figure 22 Graph depicting InfraredSolarModules dataset versus real world solar panel surface anomaly proportions 54
Figure 23 Visual representation of a single and multiple cell damage occurrences on a soral panel and some images from our database 55
Figure 24 Visual representation of a single and multiple bypass diode damage on a soral panel and some images from our database 55
Figure 25 Visual representation of a single and multiple hot spots on a soral panel and some images from our database 56
Figure 26 Visual representation of shadowing on a soral panel and an image from our database 57

Figure 27 Visual representation of an offline module and an image from our database 57
Figure 28 Visual representation of soiling on a soral panel and an image from our database 58
Figure 29 Visual representation of a confusion matrix 60
Figure 30 Mathematical and visual representation of the accuracy metric 61
Figure 31 Mathematical and visual representation of the precision metric 61
Figure 32 Mathematical and visual representation of the recall metric 62
Figure 33 Mathematical and visual representation of the F1 score metric 62
Figure 34 Some examples of the manual annotation progress 66
Figure 35 On the left, there are 2 random images of solar panels containing the cell class anomaly. On the left, there are 2 random images of solar panels containing the cell class anomaly. We depict those 4 images to highlight the similarity between these 2 surface anomalies on the infrared spectrum 68
Figure 36 One of the initial images that we sourced from the web. The panel in this image has easily recognizable damage spots. 71
Figure 37 The 4 images that we generated using the data augmentation functions flip and rotate. These images were created after the downscaling and grayscaling of the original image. 71
Figure 38 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the small dataset for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01 73
Figure 39 Classification loss function graph for the training of the YOLOv8 nano model for 120 epochs on the small dataset 73
Figure 40 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the small and more inclusive datasets for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01 74
Figure 41 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the more inclusive and modified datasets for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01 75
Figure 42 Line graph of the F1 and mAP50-90 scores Rates for different batch sizes on the complete dataset for the YOLOv8 small model. The model's number of epochs is 100 and the model's learning rate is 0.01 77
Figure 43 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the complete dataset for the YOLOv8 small model. The model's number of epochs is 100 and the model's batch size is 16 78
Figure 44 A cluster column diagram of the various F1 scores of our models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above 79

Figure 45 A cluster column diagram of the various mAP50-90 scores of our models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above 80
Figure 46 Figure 38 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the binary dataset for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01 82
Figure 47 Validation classification loss function graph for the binary training cycle of the model YOLOv8 nano 83
Figure 48 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the binary dataset for the YOLOv8 nano model. The model's number of epochs is 100 and the model's batch size is 16 84
Figure 49 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the binary dataset for the YOLOv8 nano model. The model's number of epochs is 100 and the model's batch size is 32 84
Figure 50 A cluster column diagram of the various F1 scores of our binary models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above 86
Figure 51 A cluster column diagram of the various mAP50-90 scores of our binary models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above 87

Table of Tables

Table 1 Studies of detecting the defects of solar cells using a deep learning approach 27
Table 2 Comparison matrix between some of the different optimizers that were available to us. The YOLOv8 model has a standard list of modifiers that can be applied to it. 76
Table 3 Comparison matrix between the most robust iteration of the YOLOv8 model we have trained and the YOLONAS model trained for 100 epochs 81
Table 4 Comparison matrix between the binary YOLOv8 models that were trained for 80 and 40 epochs each 83
Table 5 Comparison matrix between the true detection rates of our most robust multi-class classification model in validation and testing 89
Table 6 Comparison matrix between the true detection rates of our most robust binary identification model in validation and testing 90
Table 7 Studies of detecting the defects of solar cells using a deep learning approach, including our 2 tested models, which are highlighted 94

Table of Appendix Tables

Table 8 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 20 epochs 106
Table 9 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 20 epochs 107
Table 10 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 50 epochs 107
Table 11 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 50 epochs 107
Table 12 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 100 epochs 107
Table 13 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 100 epochs 108
Table 14 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 120 epochs 108
Table 15 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 120 epochs 108
Table 16 Normalized confusion matrix of the YOLOv8 large model trained on a smaller dataset for 20 epochs 108

Table 17 Testing metrics of the YOLOv8 large model trained on a smaller dataset for 20 epochs 109
Table 18 Normalized confusion matrix of the YOLOv8 nano model trained on an inclusive dataset for 80 epochs 109
Table 19 Testing metrics of the YOLOv8 nano model trained on an inclusive dataset for 80 epochs 109
Table 20 Normalized confusion matrix of the YOLOv8 nano model trained on an inclusive dataset for 100 epochs 110
Table 21 Validation metrics of the YOLOv8 nano model trained on an inclusive dataset for 100 epochs 110
Table 22 Normalized confusion matrix of the YOLOv8 nano model trained on a modified dataset for 80 epochs 110
Table 23 Validation metrics of the YOLOv8 nano model trained on a modified dataset for 80 epochs 111
Table 24 Normalized confusion matrix of the YOLOv8 nano model trained on a modified dataset for 100 epochs 111
Table 25 Validation metrics of the YOLOv8 nano model trained on a modified dataset for 100 epochs 111
Table 26 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 80 epochs, AdamW optimizer, batch size=16, learning rate=0.01 112
Table 27 Validation metrics of the YOLOv8 small model trained on the complete dataset for 80 epochs, AdamW optimizer, batch size=16, learning rate=0.01 112
Table 28 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW, batch size=16, learning rate=0.01 113
Table 29 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW, batch size=16, learning rate=0.01 113
Table 30 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, Adam optimizer, batch size=16, learning rate=0.01 113
Table 31 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, Adam optimizer, batch size=16, learning rate=0.01 114
Table 32 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, SGD optimizer, batch size=16, learning rate=0.01 114
Table 33 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, SGD optimizer, batch size=16, learning rate=0.01 114
Table 34 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, RAdam optimizer, batch size=16, learning rate=0.01 115
Table 35 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, RAdam optimizer, batch size=16, learning rate=0.01 115

Table 36 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.01 115
Table 37 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.01 116
Table 38 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.01 116
Table 39 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.01 116
Table 40 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.01 117
Table 41 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.01 117
Table 42 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.001 117
Table 43 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.001 118
Table 44 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0001 118
Table 45 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0001 118
Table 46 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.005 119
Table 47 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.005 119
Table 48 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0005 119
Table 49 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0005 120
Table 50 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.001 120
Table 51 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.001 120
Table 52 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.001 121
Table 53 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.001 121
Table 54 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs with, AdamW optimizer, batch size = 128, learning rate = 0.001 . 121

Table 55 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs with, AdamW optimizer, batch size = 128, learning rate = 0.001 122
Table 56 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.0001 122
Table 57 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.0001 122
Table 58 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.0001 123
Table 59 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.0001 123
Table 60 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.0001 123
Table 61 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.0001 124
Table 62 Normalized confusion matrix of the YOLONAS small model trained on the complete dataset for 50 epochs, AdamW optimizer, batch size=16, learning rate=0.01 . 124
Table 63 Normalized confusion matrix of the YOLONAS small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.01 125
Table 64 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.01 125
Table 65 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.01 125
Table 66 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 16, learning rate = 0.01 126
Table 67 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 16, learning rate = 0.01 126
Table 68 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 60 epochs, batch size = 16, learning rate = 0.01 126
Table 69 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 60 epochs, batch size = 16, learning rate = 0.01 126
Table 70 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 50 epochs, batch size = 16, learning rate = 0.01 127
Table 71 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 50 epochs, batch size = 16, learning rate = 0.01 127
Table 72 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 40 epochs, batch size = 16, learning rate = 0.01 127
Table 73 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 40 epochs, batch size = 16, learning rate = 0.01 127

Table 74 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 30 epochs, batch size = 16, learning rate = 0.01 128
Table 75 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 30 epochs, batch size = 16, learning rate = 0.01 128
Table 76 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 20 epochs, batch size = 16, learning rate = 0.01 128
Table 77 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 20 epochs, batch size = 16, learning rate = 0.01 128
Table 78 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.001 129
Table 79 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.001 129
Table 80 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.0001 129
Table 81 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.0001 129
Table 82 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.01 130
Table 83 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.01 130
Table 84 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.001 130
Table 85 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.001 130
Table 86 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.0001 131
Table 87 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.0001 131
Table 88 Normalized confusion matrix of YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 32, learning rate = 0.0001 131
Table 89 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 32, learning rate = 0.0001 131
Table 90 Normalized confusion matrix of the YOLOv8 nano model, that was trained for multi class classification tested on the 10% of the total database 132
Table 91 Confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the 10% of the total database 132
Table 92 Normalized confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the 10% of the total database 132

Table 93 Confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the augmented database 133
Table 94 Normalized matrix of the YOLOv8 nano model, that was trained for binary classification tested on the augmented database 133

CHAPTER 1 INTRODUCTION

As the global market for solar energy has risen ever so drastically in the last few years it has become apparent that better monitoring of solar panels is needed. Since its introduction, one of the most promising markets for the production of clean and renewable energy is photovoltaic energy [1]. PV modules experience thermo-mechanical loads during manufacturing and subsequent life stages leading to the origination of the appearance of different defects [2], which are responsible for the reduction of their efficiency, as well as their durability and reliability. These defects can also result in power loss and module degradation. Consequently, defect detection in PV modules has had substantial attention from researchers during the last years. An assortment of methods has been developed in order to effectively monitor the condition of solar panels. Visual methods include electroluminescent, infrared or RGB imaging in order to detect faults on the surface of the panels. However, examination of the panel's electrical performance, such as current or voltage measurements, are also deployed. Conventional methods, such as the I-V curve test and/or manual infrared thermography, are expensive and time-consuming ways that cannot be used in the large photovoltaic sites of today to identify malfunctioning modules or cells inside a plant [1]. Monitoring is vital when it comes to maximizing energy production and ensuring operational safety. This work is focused on detecting and classifying faults on the surface of solar panels using a combination of drone imagery and convolutional neural networks. The model produced detects whether an anomaly is or is not present on a solar panel and furthermore it can pinpoint and name the type of fault that riddles the panel. All the images used in this work were taken from an infrared camera mounted on an unmanned aerial vehicle [3].

1.1 UNMANNED AERIAL VEHICLES (UAVs)

When we refer to UAVs we talk about an aircraft without any crew on board, these vehicles are also usually referred to as drones. Regarding their operation mode, it is a bit different than aerial vehicles with crew, since they can be operated from the ground in real time, or they can be programmed to become fully automated to follow a specific route. The earliest UAV designs were developed during the mid 20th century and their use was purely militaristic [4]. In the modern age, these vehicles have helped individuals and corporations in aerial photography, agriculture, surveillance, disaster relief, firefighting, and more. When it comes to solar farm monitoring, UAVs provide a fast and

efficient way of extracting information about the surface of multiple panels at once. They can be especially effective when it comes to solar farms located in remote areas that are difficultly accessible by human technicians. Advancements in the camera lens industry led to cheaper and more efficient mounted cameras on UAVs, and thus, the vast implementation of UAVs with mounted cameras on all the work fields mentioned above. This paves the way for more inclusion of artificial intelligence models in surface anomaly detection, since the high resolution provided by these advancements will make it easier for convolutional neural networks to perform object detection and pattern recognition on the visible or infrared light spectrums. Multicopters, which employ rotary blades to provide lift, are the most suitable type of unmanned aerial vehicle equipment for thermographic investigations due to their stability and ease of usage. They can be categorized based on the number of motors (tricopters, quadcopters, hexacopters, and octocopters) [5]. Quadcopters are the most common type on the market.

1.2 INFRARED THERMOGRAPHY vs ELELECTOLUMINESCENT IMAGERY

In this thesis, we will focus on infrared images of photovoltaic panels and their potential to be used as data for our model and aid in the detection and classification of faults present on the surface of these panels. This does not mean that we shall underline the other methods of detecting faults and anomalies on solar panels. If one were to look at the research done on detection of surface faults of solar panels, they would soon realize that the majority of the papers published revolve around the use of electroluminescent images of polycrystalline panels [6]. In PV modules, EL is particularly useful for identifying cell cracks, which show up in the EL image as black lines on the solar cell, in addition to broken connections or other process failures. Since not every flaw found causes a temperature rise, outdoor EL can be used in alongside IRT to provide details about defect patterns that may not always be available through infrared thermography (IRT). Thus, occasionally it is possible to resolve some faults more precisely with the high resolution of the EL photos than with the IRT images. Therefore, it is advised to use both approaches in combination to find as many flaws as you can [1].

1.3 ARIEAL INFRARED THERMOGRAPHY (AIRT)

IRT inspections of PV systems were formerly carried out using handheld IRT cameras on the ground or on lifting platforms. In addition to being extremely labor-intensive and time-consuming, this technique is dependent on human labor and skill and is unable to offer quick and precise coverage of a power plant of considerable size. The precision of the inspection is therefore vulnerable to human mistake. Combining aerial technologies like UAVs with the IRT sensor could be one way to solve the issue. This process, called AIRT, makes IRT inspection more cost-effective and can therefore be used for large-scale PV plants or roof-mounted PV systems with restricted access [7][8]. Although this approach has demonstrated promise in recent times, there is still room to grow and streamline its application in order to revolutionize PV plant monitoring protocols going forward and enhancing and streamlining routine inspections of PV power plants [9]. However, for this approach to reach its most potential, it must be used in conjunction with automation techniques and technologies, like automated route planning and flaw detection. Aerial IRT makes use of UAV-mounted visual and IRT cameras. The apparatus offers accurate and real-time imagery, making the examination process quick. Many defect types, including cell cracks, corrosion spots, broken cells, hot spots, snail trails, discoloration, soiling, disconnected modules, shadowing, soldering mistakes, vegetation coverage can be found using this approach [1][3][10]. AIRT can be carried out from various heights and directions to find particular problems or deficiencies, depending on the inspection goal and the layout of the PV plant. Three steps make up the AIRT process: gathering imagery, evaluating data, and taking corrective action. In order to create an image database that covers every module in the system, the UAV must fly a predetermined path to take successive pictures or videos over the site during the acquisition process. Flight plans must take the battery capacity of the drone into consideration. The weather, velocity of the wind, and sunlight reflection must all be watched throughout the flight because they can have an impact on the readings as well and, in turn, the quality of the AIRT images. Furthermore, self-shading, blurry pictures, and other reflections must be prevented by controlling the UAV's airspeed as well as the orientation and angle of the IRT sensor. An AIRT shall be carried out on days that are clear, sunny, and cloudless, with a minimum of 600 W/m2 of irradiance on the plane of the PV array that is being inspected. To maximize coverage and prevent drift during the flight, the flight path and velocity should be prearranged [11][12].

Figure 1 Image of a UAV performing AIRT

1.4 CONVOLUTIONAL NEURAL NETWORKS (CNNs)

Without neural networks, the process of classifying images would not be feasible. One of the many deep learning algorithms used in fields like object detection and pattern recognition is the convolutional neural network (CNN). They offer functions including quick processing and the capacity to learn by examining a collection of inputs [13]. CNNs are an especially advanced technique for classifying images and were inspired by the structure of the human visual system [14]. If there is a sizable collection of labeled images that represent various categories, they are simple to train. Each of the layered sequences has a distinct purpose in the input signal's propagation. Neural layers can be classified into three primary types: fully connected layers, pooling layers, and convolutional layers.

• Convolutional layers: these layers take care of taking characteristics out of the input images.

• Pooling layers: these layers lower the input volume's spatial dimensions after the convolutional layers, which lessens the network's computational load.

• Fully connected layers: function like a traditional neural network in doing the final classification of the photos.

We will further analyze CNNs later in this thesis, in chapter 2 more specifically.

1.5 PAST WORK ON THE FIELD

As we mentioned before, a deep learning approach has emerged as a powerful tool to solve problems related to visual computing and recognition of patterns, this includes object detection and image classification. In this section we look at multiple models that were trained for such purposes The CNN's outstanding performance was indicated by [15]. The results showed an accuracy of 88.42%, outperforming the SVM model by 6%. This model achieved satisfactory results, but the false positive rate remains high, and the classification performance is not up to expectations. This could be caused by the sameness of the defect characteristics and the complicated background of solar cells. In [16], Karimi et al. evaluated the automation data analysis pipeline for solar defects recognition using EL images. A comparison of three models (SVM, RF, and CNN) was conducted. The CNN model outperformed other methods with an accuracy rate of 99.42%. However, the result of the application is a binary classification. The Electroluminescence (EL) images were dominant across the reviewed studies in terms of the image types. This makes the use of IRT images a rarer approach. Overall, most of the models achieve an accuracy rate greater than 90%. However, more than 60% of the reviewed models can do two-class classification (Binary classification) to determine whether the case is faulty or healthy. It was noticed that the models excelling at multi-class classification are less than those designed for binary classification. Moreover, the complexity of the two-class classification models requires more computational resources. Another study that we must focus our attention on was conducted in 2021 [17]. This team of researchers used the same publicly available dataset that we have proposed using. Since they had 20000 images available to them thus, they did not use data augmentation. They approached a task remarkably similar to ours, to propose a general CNN framework to classify distinct categories of faults and degradation modes. CNN's architecture is thoroughly detailed in the research paper as well as the training methodology that the researchers used. The model was able to correctly identify 92% of healthy PV modules and 93% of damaged modules. A research paper that was written in China in 2021 took a novel approach using the yolov5 model in combination with the ResNet model to detect faults in photovoltaic panel arrays [18]. Their approach was unique because they used a combination of IRT images and normal RGB images to extract information used in the assessment of errors. YOLOv5 is used for image segmentation, the defect detection is handled by the ResNet model. Compared to other neural networks like RCNN series, SSD, the proposed approach is much faster with the same segmentation result. The average detect speed is 72 times faster than Fast-RCNN after testing in 100 images validation set. The accuracy loss of our approach is less than 1.7%, and the speed is at least doubled, and it is argued that over 90% accuracy means

almost perfect result in the field of image segmentation. In conclusion it is stated that the YOLOv5 algorithm dramatically boosts this approach's detection efficiency.

Table 1 Studies of detecting the defects of solar cells using a deep learning approach

	Model	No. of para-meters	Number of layers	Type of image	Performance criteria	Remarks
[82]	VGG-16	0.38 M	21	EL	7.7% (balanced error), 92.3% (balanced accurate)	Considers effects of augmentation and oversampling.
[83]	DBN	4.7M	4	EL	NA	Time-consuming
[84]	CNN	1.1 M	6	EL	98.4%	Small datasets
[85]	CNN	101.2	M 27		RGB 94.3%	Considers only visible defects, not applicable for weak scratch inspection.
[86]	CNN, SVM	34.9 M	24	EL	88.42% (CNN), 82.44% (SVM)	Processing did not distinguish the defect type
[87]	CNN	0.2 M	5	EL	99.43% (SVM), 97.46% (RF), 99.71 % (CNN)	Two-category result
[88]	CNN	2.5 M	9	EL	93.02% Two-category result,	Can suffer from the over-fitting phenomenon.
[89]	CNN	12.9 M	9	EL	Micro-crack (82%), finger-interruption (81%), break (83%)	Two-category result.
[90]	CNN with Attention network & U-net	8.1 M	35	EL	99.3%	Considers hybrid loss and incorporating CNN model with other networks
[91]	CNN (LeNet)	0.06 M	7	EL	99%	Outperforms GoogleNet
[92]	R-CNN & R-FCN	-	-	EL	98.3%	The strategy of hard negative sample mining was used.
[93]	CNN	34.1 M	8	EL	F-measure 98.46%	Fuses steerable evidence filter (SEF) with the function of structural decoupling to filter the input images.
[94]	CNN (AlexNet)	61 M	25	RGB	93.3%.	Two-category result.
[95]	U-Net & Attention mechanism	-	28	EL	IOU (69%) DICE (54%)	Solves the All black issue.
[96]	CNN (ResNet50)	23.5 M	51	EL	98.59%	Used on-field low-resolution EL images.
[97]	CNN (U-Net)	-	-	EL & C-DCR	F1-score is (89%) for dark saturation current density (j0) (82%) for series resistance (Rs)	Introduces smart labeling of defects.
[98]	CNN (ResNet-50)	5.4 M	7	EL	91%	Examines the effect of different parameterizations of the normalized Lp layer on the segmentation performance
[99]	YOLOv3	-	53	RGB	94.5%	Two-category result.

1.6 NOVELTY OF OUR THESIS

As of this day YOLOv8 has been used only once in photovoltaic fault detection in a publicly available research paper [19]. The research was conducted in the USA in 2023. The YOLOv8 model was used together with the PSO algorithm to optimize the model's parameters to achieve the best detection accuracy. The dataset that was used was made up of over 2000 EL images of solar panels, where some were faulty. In this thesis, the YOLOv8 model has been trained, validated and tested on aerial infrared images of solar panels with multiple fault categories. The YOLONAS model has also been trained and validated with the same dataset. In order for training of the models to be possible, the time-consuming annotation process was required. Manual annotation for the particular dataset has not published in any other work. The model was trained to perform binary classification or multi-class fault identification and to pinpoint each error with an indication square. Through experimentation and multiple parameter tweaking, we modified the YOLOv8 model to optimally detect faults on IRT images. No scientific work has been published, where solar panel surface faults are categorized and pinpointed using YOLOv8. Our aim is to prove that through thorough training, a single model can have different specialized versions, each one being specialized in a different detection task. The testing of our model was also conducted through the use of a freshly created dataset from images found in the web, with the use of data augmentation.

CHAPTER 2 THEORY

2.1 Artificial Intelligence (AI)

At the start of our theoretical presentation, we shall make a clear distinction between some definitions that are often confused, those being Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL). Artificial intelligence (AI) refers to the intelligence that is possessed by computers or software, as opposed to the intellect that is possessed by living organisms, mostly humans. In the discipline of computer science, it is a subfield that focuses on the creation and investigation of intelligent machines. AIs are a term that can be used to refer to such machines. In the fields of industry, government, and science, artificial intelligence technology is frequently utilized. Some of the most prominent uses include enhanced web search engines, recommendation systems, self-driving cars, and generative tools (such as ChatGPT and DALI) [20]. It was Alan Turing who was the first person to carry out significant research in the topic that he referred to as machine intelligence [21].

2.2 MACHINE LEARNING

Machine learning (ML) is a branch of artificial intelligence that focuses on creating and analyzing statistical algorithms capable of learning from data and applying that knowledge to new, unseen data. This enables ML systems to complete tasks without the need for explicit instructions [22]. Machine learning techniques have been utilized in several domains such as natural language processing, computer vision, speech recognition, email filtering, agriculture, and medicine [23][24]. Machine learning, commonly referred to as ML, is widely recognized for its use in solving business challenges using a technique called predictive analytics. While not all machine learning techniques rely on statistics, computational statistics play a crucial role in providing the methodologies used in this discipline. Mathematical optimization methods form the mathematical basis of machine learning. Arthur Samuel, an IBM employee, and innovator in the field of artificial intelligence, introduced the term "machine learning" in 1959. Deep Learning is a specific subset of Machine Learning.

2.3 DEEP LEARNING

Deep learning is a branch of machine learning that focuses on using artificial neural networks (ANNs) to learn and represent data. The term "deep" refers to the utilization of numerous layers inside the network. The methods employed can fall into three categories: supervised, semi-supervised, or unsupervised [25]. Deep-learning architectures, such as deep neural networks, deep belief networks, recurrent neural networks, convolutional neural networks, and transformers, have been utilized in various domains, including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, climate science and material inspection. In these fields, deep-learning architectures have achieved outcomes that are comparable to, and in certain instances, superior to human expert performance [26][27]. In the field of image processing, lower layers are responsible for detecting edges, while higher layers are responsible for recognizing more complex ideas that are meaningful to humans, such as numerals, characters, or faces. In order for the network to develop the ability to recognize things and make accurate decisions, it must be exposed to a wide range of samples. Furthermore, deep learning requires a substantial number of computational resources. However, the introduction of high-performance GPUs has greatly decreased the time required to train a network [28].

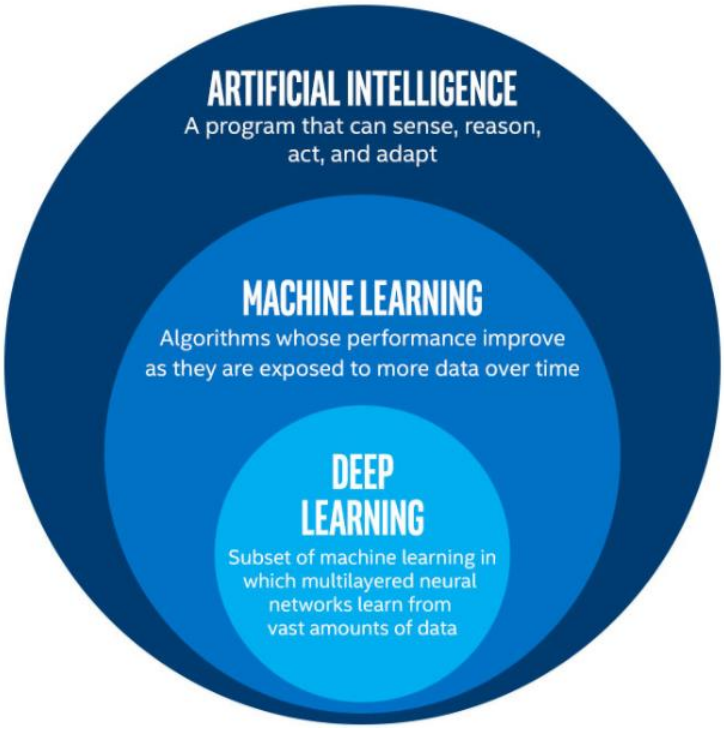

Figure 2 Visualization of the differences between AI, ML and DL

2.4 Convolutional Neural Network (CNN)

CNN is a feedforward neural network capable of extracting features from data using convolutional structures. CNN does not require manual feature extraction, unlike conventional approaches [29][30][31]. The design of CNN is influenced by visual perception [32]. A biological neuron is analogous to an artificial neuron. In CNN, kernels operate as receptors detecting different properties. Activation functions mimic the process where only neural electric signals beyond a specific threshold are passed to the next neuron. Loss functions and optimizers are tools created to train convolutional neural networks to meet desired learning outcomes. CNN has numerous advantages compared to fully connected (FC) networks. [33]

1) Neurons have limited connections to a small number of neurons in the preceding layer, rather than being connected to all neurons. This helps reduce parameters and accelerate convergence.

2) Weight sharing involves several connections using the same weights, leading to a reduction in parameters.

3) Dimension reduction, or downsampling, involves using a pooling layer to decrease an image's size by exploiting local correlations, which helps minimize data volume while preserving key details. It can decrease the number of parameters by eliminating insignificant aspects. CNN is considered one of the most representative algorithms in the deep learning field due to its three appealing properties.

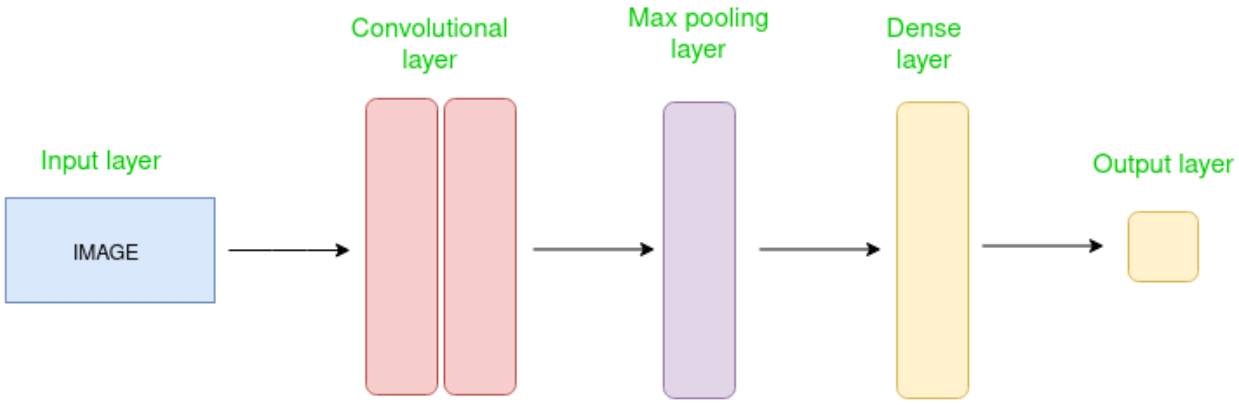

Figure 3 Simplified visualization of a convolutional neural network structure

2.4.1 CONVOLUTION LAYER

To provide further clarification, in order to construct a CNN model, it is typically necessary to adjust four components. These are the convolution kernel's size, the padding that will be used, the number of image channels and our choice of stride.

Kernels are compact filters that are used to process image data by sliding across its width and height, calculating the dot product between the pixel values of the image and the values of the kernel. This process generates a feature map that represents the output of the filter. In the context of convolution, the outputs can be referred to as feature maps. When we set a convolution kernel to a particular size, we will lose information on the border of the image [34]. Consequently, padding is implemented in order to enlarge the input with zero value, which can enable the size to be adjusted in an indirect manner. This is called zero padding. Copying the value of the edge pixel is one other frequently used technique called replicate padding. The number of image channels that are used can be either three or one. One channel is utilized for grayscalled images, whereas 3 channels are utilized for RGB images. We refer to the size of the steps that the kernel makes as it traverses through the input image as stride. Stride is utilized in order to accomplish the control of the density of the convolution. There is a correlation between the size of the stride and the density.

Following the process of convolution, feature maps are comprised of a substantial number of features, which make them susceptible to the overfitting problem [35]. Because of this, the elimination of redundant information is proposed through the use of pooling [36], also known as downsampling. This process is executed in the pooling layers of the convolution neural network.

2.4.2 POOLING LAYER

The pooling layers are quite significant in CNN training. After performing the convolution process, we construct a pooling layer in order to extract the values that are the most informative. When it comes to neural network training, pooling can serve as a generalizer of the lower-level data [33]. For the purpose of obtaining the contents of each window, this method makes use of a number of different statistical functions. The maximum pooling technique and the average pooling technique are the two most popular forms of pooling strategies. In order to extract useful information, the max pooling technique makes use of the max function, whereas the average pooling technique makes use of the statistical mean function. In general, a pooling function

reduces the output features by performing nonlinear computing and reducing the number of parameters. When the pooling algorithm is put into action, it lowers the resolution of the feature maps that cover the local neighborhood of the layer that came before it.

When the complexity of the images is taken into consideration, the number of convolutional and pooling layers may be raised in order to capture low-level information even further. However, this will result in an increase in the amount of computer power required.

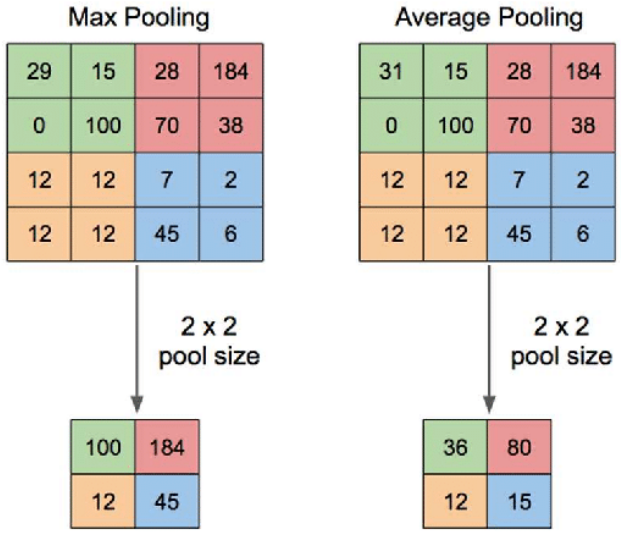

Figure 4 Visualization of the 2 different pooling processes

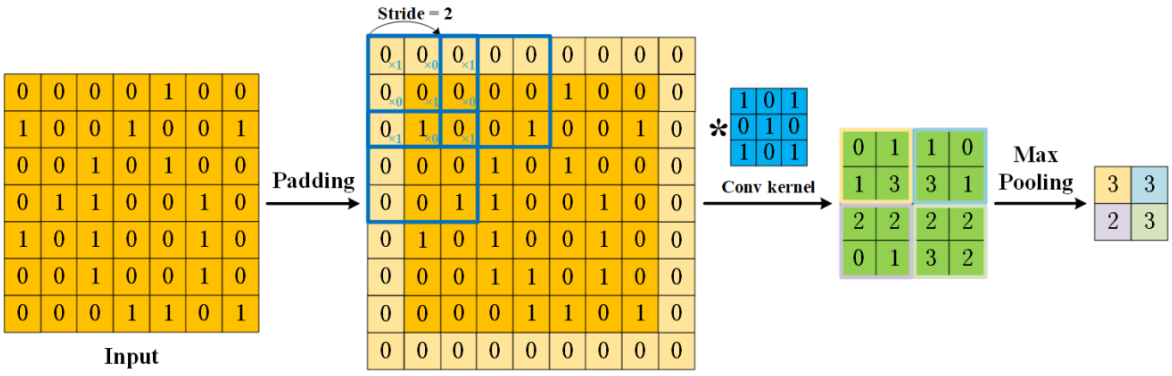

Figure 5 The process that a CNN goes through in the convolutional and pooling layers

2.4.3 FULLY CONNECTED LAYER

A Fully Connected (FC) layer, sometimes referred to as a dense layer, is a layer utilized in neural networks where every neuron or node from the preceding layer is connected to every neuron in the current layer. The term "fully connected" is used due to the presence of total connectivity. FC layers are commonly located at the latter part of a neural network structure and are accountable for generating the final output predictions. In CNNs, FC layers are positioned after the convolutional and pooling layers. They are employed to convert the two-dimensional spatial arrangement of the data of the feature maps into a one-dimensional vector and analyze this data for tasks such as classification. The number of neurons in the last FC layer is equal to the number of output classes in a classification task. For example, in an 8-class classification task, the final FC layer would consist of 8 neurons, each producing a score corresponding to one of the classes.

2.5 ACTIVATION

A particular function known as an activation function is utilized in artificial neural networks for the simple purpose of converting an input signal into an output signal [37]. This output signal is then employed as an input to the subsequent layer in the stack. An artificial neural network is constructed by first calculating the sum of products of inputs and the weights that correspond to them, and then applying an activation function to the resulting output in order to obtain the output of that particular layer, which is then used as the input for the subsequent layer below it.

The majority of the time, a neural network that does not have an activation function performs as a linear regression model, with restricted performance and power. A neural network should be able to not only learn and compute a linear function, but also execute tasks that are more complex than that, such as analyzing complex sorts of data, such as photos, videos, audio, speech, text, and so on.

Some important examples of activation functions that we shall mention are the following [37]:

1. Binary Step Function

2. Linear

3. Sigmoid

4. Tanh

5. ReLU

6. Leaky ReLU

7. Parametrized ReLU

8. Exponential Linear Unit

9. Swish

10. SoftMax

2.5.1 BINARY STEP

The Binary Step Function is the most basic activation function and may be easily constructed in Python using simple if-else statements. Binary activation functions are typically employed while developing a binary classifier. Binary step function is not suitable for multiclass classification. This function can be expressed mathematically as:

$$f(x) = 1\,,\ x \geq 0$$
$$f(x) = 0\,,\ x < 0$$

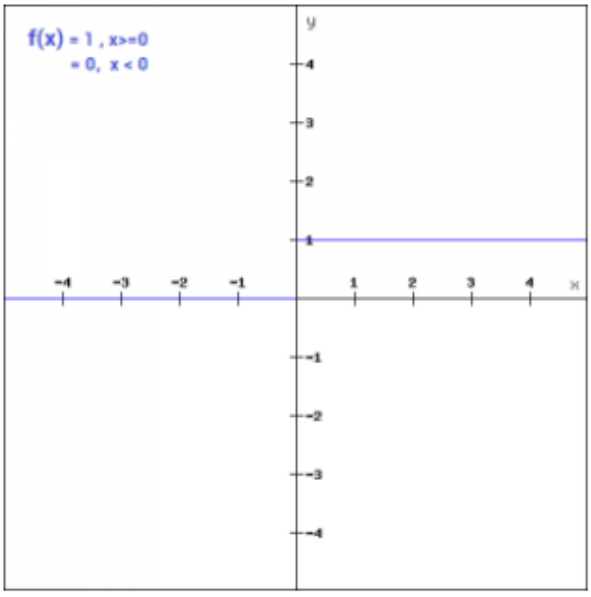

Figure 6 Plot of the binary step activation function

2.5.2 LINEAR

The linear activation function increases in direct proportion to the input. Using a linear function in a neural network does not provide much benefit as it results in the same gradient value for every iteration, hindering error improvement. A network lacks the ability to recognize intricate patterns within the data when this function is applied. Linear functions are best suited for jobs that need interpretability and simplicity. This function can be defined mathematically as:

$$f(x) = ax$$

Variable a can be assigned any constant value selected by the user.

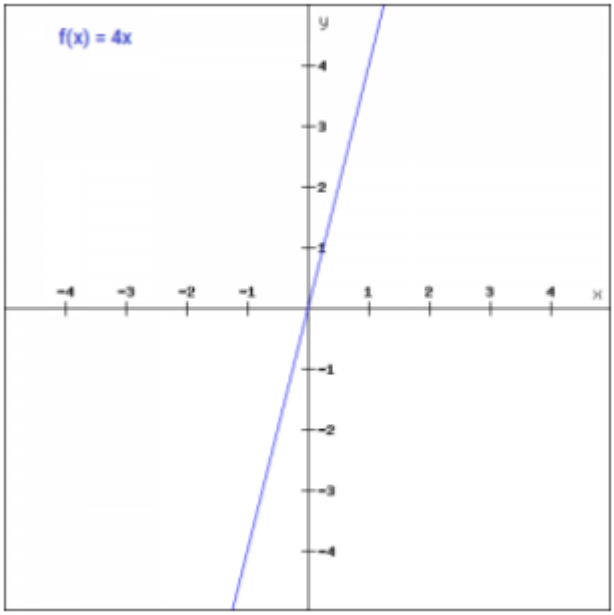

Figure 7 Plot of the linear activation function

2.5.3 SIGMOID

It is the most commonly utilized activation function due to its non-linear nature. The sigmoid function maps values to the interval between 0 and 1. The function can be defined as:

$$f(x) = \frac{1}{e^{-x}}$$

The sigmoid function is continuously differentiable and exhibits a smooth S-shaped curve. This function is asymmetric about zero, resulting in all output values of neurons having the same sign. To enhance this matter, consider scaling the sigmoid function.

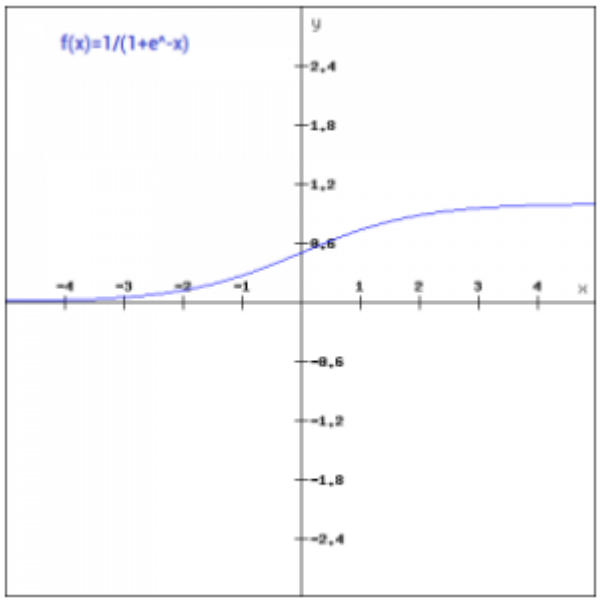

Figure 8 Plot of the sigmoid activation function

2.5.4 TANH

The Tanh function is akin to the sigmoid function. However, it displays symmetry around the origin of the axes. This leads to varied outputs from previous layers, which are then used as input for the subsequent layer.

The function can be defined as:

$$f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

The hyperbolic tangent function is both continuous and differentiable, with values ranging from -1 to 1. The gradient of the tanh function is steeper than that of the sigmoid function. Tanh is favored over the sigmoid function due to its unrestricted gradients and zero-centered nature.

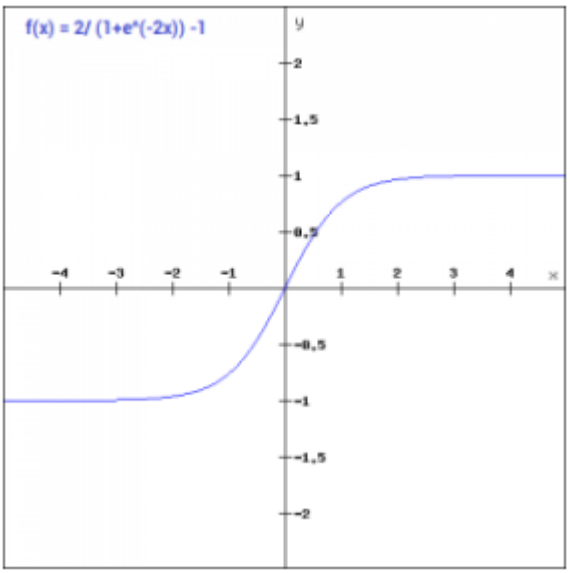

Figure 9 Plot of the tanh activation function

2.5.5 RELU

ReLU is short for rectified linear unit, a non-linear activation function commonly utilized in neural networks. An advantage of utilizing the ReLU function is that not all neurons are stimulated simultaneously. This means that a neuron will only be silenced if the outcome of the linear transformation is zero. The mathematical definition is:

$$f(x) = \max(0\,,\, x)$$

ReLU is more efficient than other functions because it activates only a certain number of neurons at a time, instead of all neurons simultaneously.

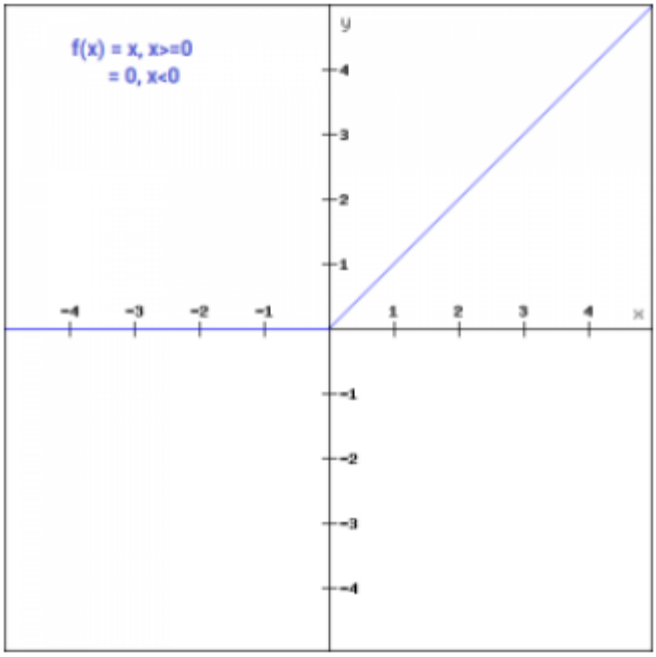

Figure 10 Plot of the relu activation function

2.5.6 LEAKY RELU

Leaky ReLU is an enhanced version of the ReLU function that assigns a small linear value to negative x values instead of setting it to zero. The mathematical expression is as follows:

$$f(x) = 0.01x,\ x < 0$$

$$f(x) = x,\ x \geq 0$$

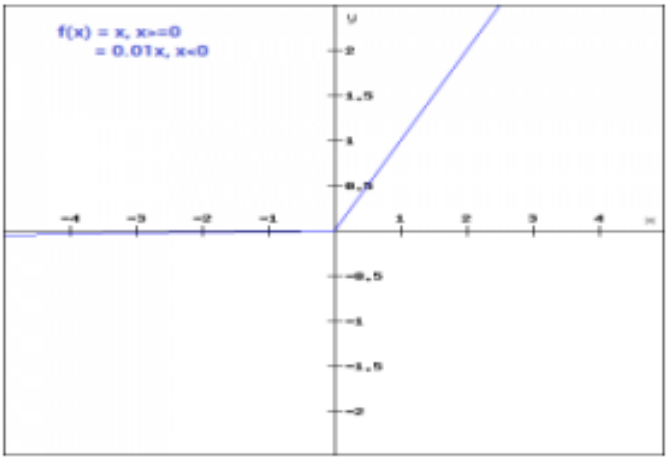

Figure 12 Plot of the leaky relu activation function

2.5.7 PARAMETRIZED RELU FUNCTION

It is a variation of Rectified Linear Unit activation function that offers improved performance and a little modification. The issue of the ReLU gradient being 0 for negative x values is addressed by introducing a new parameter called the slope (a in this example) for the negative section of the function. The function is defined as follows:

$$f(x) = x,\ x \geq 0$$

$$f(x) = ax,\ x < 0$$

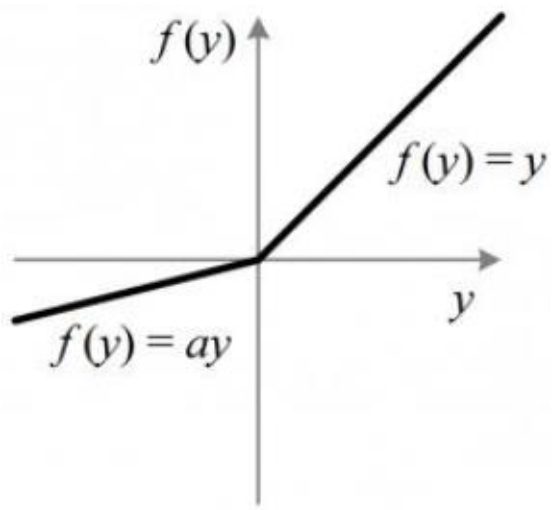

Figure 11 Plot of the parametrized relu activation function

2.5.8 ELU

ELU, short for Exponential Linear Unit, is a variation of Rectified Linear Unit activation function. ELU also introduces a parameter called slope for the negative values of x. The negative values are defined using a logarithmic curve. The function f(x) is defined as:

$$f(x) = x,\ x \geq 0$$

$$f(x) = a(e^x - 1),\ x < 0$$

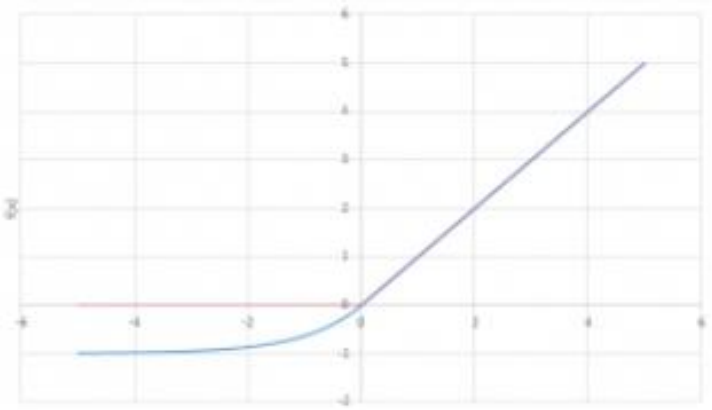

Figure 12 Plot of the ELU activation function

2.5.9 SWISH

The Swish function is a recently developed activation function by researchers at Google. The unique characteristic of the Swish function is its non-monotonic nature, where the function's value can drop despite increasing input values. Swish function can outperform the ReLU function in some scenarios.

It is expressed mathematically as:

$$f(x) = \frac{x}{1 - e^{-x}}$$

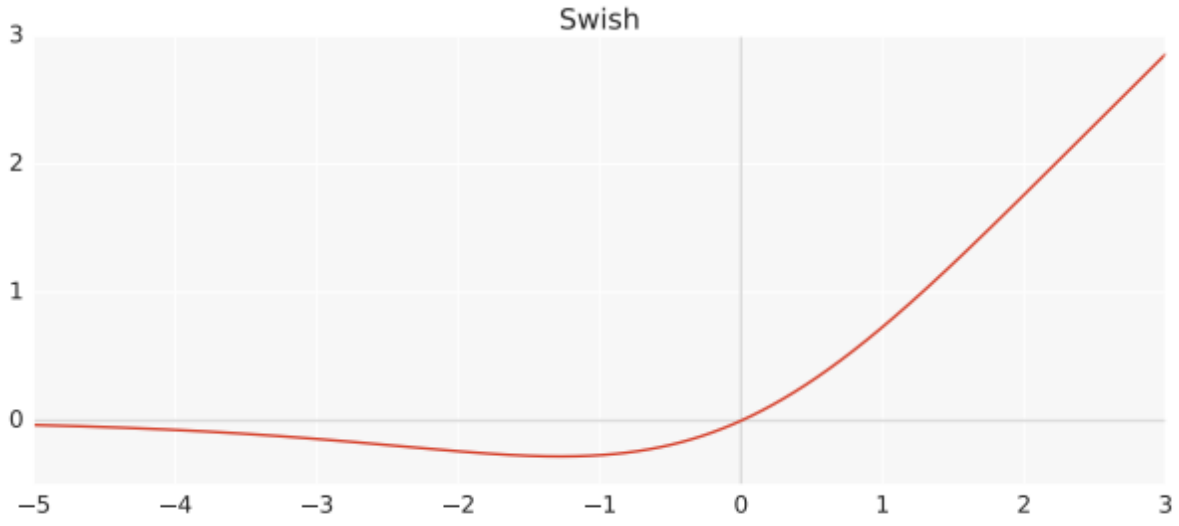

Figure 13 Plot of the swiss activation function

2.5.10 SOFTMAX

The softmax function is a composite of several sigmoid functions. The sigmoid function outputs values between 0 and 1, which can be interpreted as the probabilities of data points belonging to a specific class. This activation function is suitable for multiclass classification tasks, unlike sigmoid functions which are typically employed for binary classification. The function calculates the probability for each data point in every class. When constructing a network or model for multi-class classification, the output layer will contain a number of neurons equal to the number of classes in the target.

It is expressed mathematically as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; for\; j = 1, \ \ldots, K$$

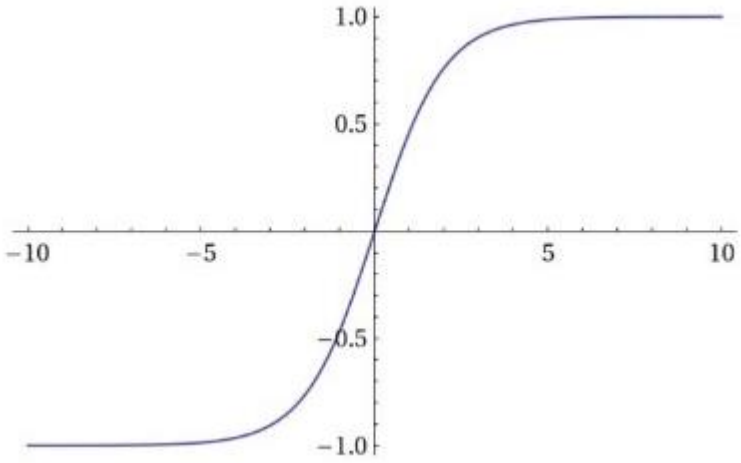

Figure 14 Plot of the Softmax activation function

2.6 LOSS FUNCTIONS

The loss function, also known as the cost function, is utilized to compute the discrepancy between the anticipated value and the actual value. The loss function often serves as the learning criterion for the optimization issue. The loss function is utilized in Convolutional Neural Networks (CNNs) to address both regression and classification issues, with the objective of minimizing the loss function. Some commonly utilized loss functions are mean absolute error (MAE), mean square error (MSE) and cross entropy. Here we will briefly touch upon these loss functions as well as some more niche loss functions

1) Regression Loss Function: In Convolutional Neural Networks (CNNs), it is common to utilize Mean Absolute Error (MAE) or Mean Squared Error (MSE) as loss functions to address regression concerns. The Mean Absolute Error (MAE) calculates the average of the absolute difference between the predicted value and the actual value. On the other hand, the Mean Squared Error (MSE) calculates the average of the squared difference between them. When the training set contains numerous outliers that could potentially harm the models, Mean Absolute Error (MAE) is a superior choice compared to Mean Squared Error (MSE). Alternatively, one should take into account MSE.

2) Classification Loss Function: Convolutional Neural Networks (CNNs) employ several loss functions to address classification tasks. The commonly utilized method, known as cross entropy loss, is employed to assess the disparity between the probability distribution derived from the current training and the actual distribution. This function evaluates the discrepancy between the anticipated probability and the actual output value (0 or 1) for each class and computes a penalty value based on the distance between them. The penalty is logarithmic, resulting in a lower score (0.1 or 0.2) for smaller discrepancies and a higher score (0.9 or 1.0) for greater differences.

The cross entropy loss, also known as softmax loss, is exclusively utilized in convolutional neural networks (CNNs) in conjunction with a softmax layer. The cross entropy loss function exhibits several limitations. The cross entropy loss function focuses solely on the accuracy of classification, disregarding the level of coherence within the same category or the separation between separate categories. Therefore, numerous loss functions have been suggested to address this issue.

The contrastive loss function [38] increases the separation between distinct categories and reduces the separation within the same categories. It is applicable for reducing the

dimensions in Convolutional Neural Networks (CNNs). Following the process of dimensionality reduction, the two samples that were initially similar remain similar in the feature space. Conversely, the two samples that were initially distinct continue to be different. Moreover, the contrastive loss is extensively employed in conjunction with convolutional neural networks (CNNs) for the purpose of face recognition.

Another approach is the utilization of center loss [39], which is a refinement built upon the concept of cross entropy. The objective of center loss is to emphasize the consistency of the distribution within each individual class. To achieve a balanced distribution throughout the center of the class, center loss imposes an extra restriction to minimize the disparity within each class. Center loss has been applied in conjunction with Convolutional Neural Networks (CNN) for various tasks such as face recognition [39], picture retrieval [40], person re-identification [41], speaker recognition [42], and others.

The large margin softmax loss [43] is also an alternative form of cross entropy. Its objective is dual: intraclass compression and interclass separation. The large margin softmax loss incorporates a margin between distinct classes and introduces margin regularity by considering the angle of the constraint weight matrix. The large margin softmax loss function has been applied in several applications such as face recognition [43], emotion recognition [44], and speaker verification [45].

2.7 OPTIMIZER

Optimizers are essential in the field of deep learning because they are algorithms that dynamically fine-tune the parameters of a model throughout the training process with the goal of minimizing a loss function that has been set [46]. Through the refinement of the weights and biases based on the input obtained from the data, these specialized algorithms make the learning process of neural networks easier to accomplish. Stochastic Gradient Descent (SGD), Adam, and RMSprop are three well-known optimizers in the field of deep learning. Each of these optimizers is equipped with a unique set of update rules, learning rates, and momentum strategies. These different optimizers are all geared toward the overarching goal of discovering and converging upon optimal model parameters, which ultimately results in an improvement in overall performance.

2.8 HYPERPARAMETERS

In the process of building a CNN, in addition to choosing the activation function, loss function, and optimizer, we are required to tune a large number of other hyperparameters, which have a significant impact on the performance of the model [33]. Each and every one of us is aware of the fact that there is no predetermined collection of hyperparameters that can always ensure an ideal result. Consequently, when it comes to tuning hyperparameters, having a set of rules and experience is quite important.

During the process of tuning a model, it is simple to consider searching through all of the hyperparameters to find the optimal model. When it comes to the specifics, if there are two hyperparameters that need to be tuned and each of them has five possible values, then there will be a total of 25 tests that need to be carried out. However, as the number of hyperparameters increases, the number of trials will inevitably expand exponentially. Furthermore, if we do not have any indication which hyperparameter is more significant, then it is difficult to establish which hyperparameter ought to be set first. The most important of the hyperparameters are mentioned here:

2.8.1 LEARNING RATE

In the context of updating network weights, the term "learning rate" refers to the step size. The value may be either constant or variable. Different learning rates are determined by optimization algorithms. Using a relatively big learning rate at the beginning of the training to speed up the training, using a small learning rate to promote training stability, and avoiding skipping the optimal value are the fundamental ideas behind the adjustment strategy. The goal of the adjustment strategy is to have the learning rate gradually decline with the training.

2.8.2 NUMBER OF EPOCHS

Epoch is a term that describes the number of times that the entire training set is fed into the neural network throughout the training process. In situations where the difference in accuracy between the training set and the validation set is quite small, the current epoch is more suitable. A decrease in the gap indicates that the epoch is too small, which leads to underfitting. On the other hand, an increase in the gap indicates that the epoch is too large, which leads to overfitting. The epoch should be made as large as possible at the beginning, if the computing resources are available, and then the training should be

terminated when the training loss is steady or when the training accuracy is comparable to the validation accuracy. In the event that there are restricted computing resources, it is appropriate to train as many epochs as possible.

2.8.3 BATCH SIZE

The number of samples that are provided to the model during its training is referred to as batch size. In the process of optimizing a network, a small batch size indicates that the number of samples that are input into the network is insufficient, which means that the samples are not representative. As a result, the noise increases, and the network becomes harder to converge from one point to another. A batch size that is excessively big produces a grading direction that is practically steady and makes it simple to fall into the local optimum or saddle point. Depending on the amount of memory available on the GPU and the computing core, the Batch size is typically set to the power of two. This allows the GPU to make the most of its performance capabilities. The batch size is typically set from 8 to 256.

2.9 ANALYSIS OF THE YOLO MODELS

The publication titled "YOLO" authored by Joseph Redmon et al. was presented at the CVPR 2016 conference [47]. It introduced a novel method for object detection that operates in real-time and covers the entire process from start to finish. The acronym YOLO stands for "You Only Look Once," which highlights its ability to detect objects in a single pass of the network. This is in contrast to previous methods that either relied on sliding windows followed by multiple runs of a classifier per image, or more advanced techniques that divided the task into two steps: first detecting possible object regions or proposals, and then running a classifier on those proposals.

2.9.1 APPLICATIONS

The YOLO family of models has been utilized in a vast variety of tasks across many industries. Some examples are:

• The monitoring and detection of several entities on the road, including automobiles, pedestrians [48][49], bicycles, and other barriers [50][51][52][53].

• Sports analysis [54]

• Identification categorization of diverse types of crops [55][56], pests, and diseases [57]. Assisting in implementing precise agricultural methods and automating farming procedures.

• Face detection in biometrics, security, and facial recognition systems [58][59].

• Detection of cancer tumors [60][61]

• Pill identification [62]

• The process of categorizing objects and features in satellite and aerial imagery, used to assist in creating maps, planning cities, and monitoring the environment [63][64][65][66].

• Identification and recognition of license plates, specifically referring to the process of detecting and extracting license plate information from images or videos. [67].

• Traffic sign recognition [68]

• Detecting and monitoring wildlife in order to identify species that are at risk of extinction. This information is used to protect biodiversity and manage ecosystems effectively [69].

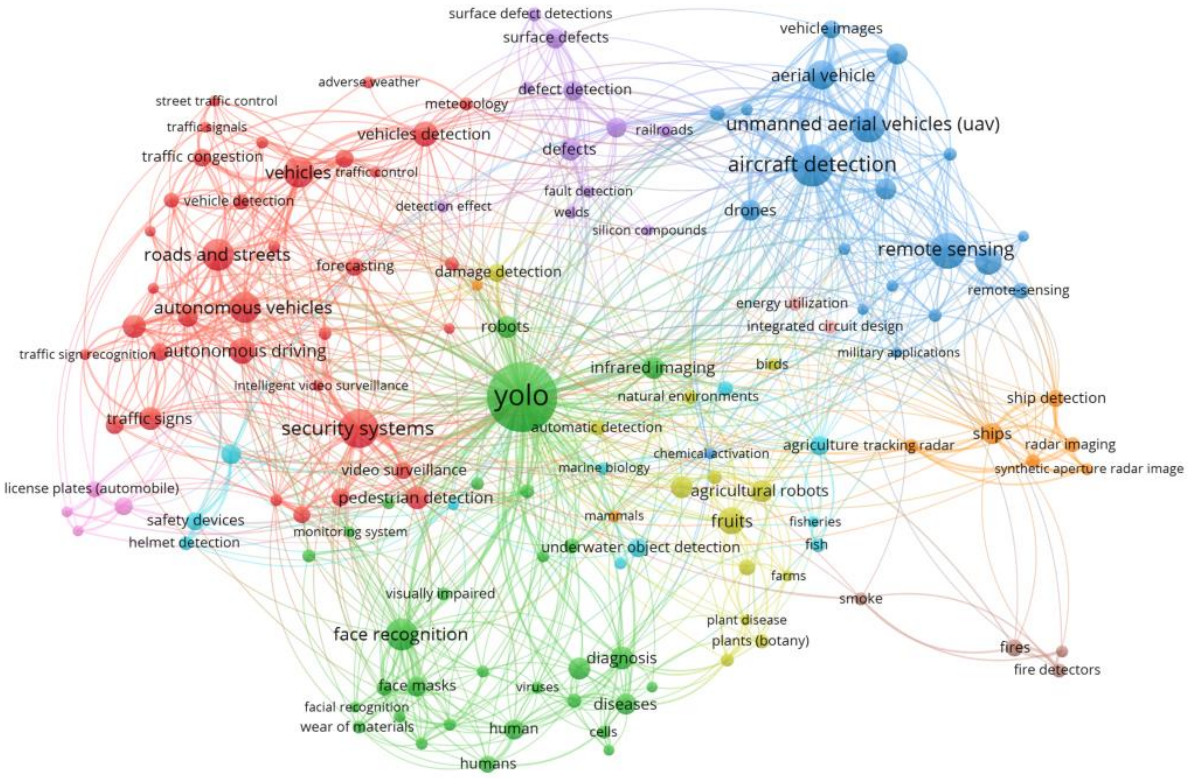

Figure 15 Bibliometric network visualization of the main YOLO Applications

2.9.2 FIRST APPROACH (YOLOv1)

YOLOv1 streamlined the object detection process by simultaneously identifying all the bounding boxes. YOLO achieves this by partitioning the input image into a grid of size S×S and making predictions for B bounding boxes of the same category, coupled with the corresponding confidence scores for C distinct categories per grid cell. The bounding box prediction comprises five values: Pc, bx, by, bh, bw. Pc is the confidence score assigned to the box, indicating the model's level of confidence in the presence of an object and the accuracy of the box. The bx and by coordinates represent the center points of the box in relation to the grid cell. The bh and bw values indicate the height and width of the box in relation to the entire image. The result of YOLO is a tensor with dimensions S × S × (B × 5 + C), which can be further processed using non-maximum suppression (NMS) to eliminate redundant detections.

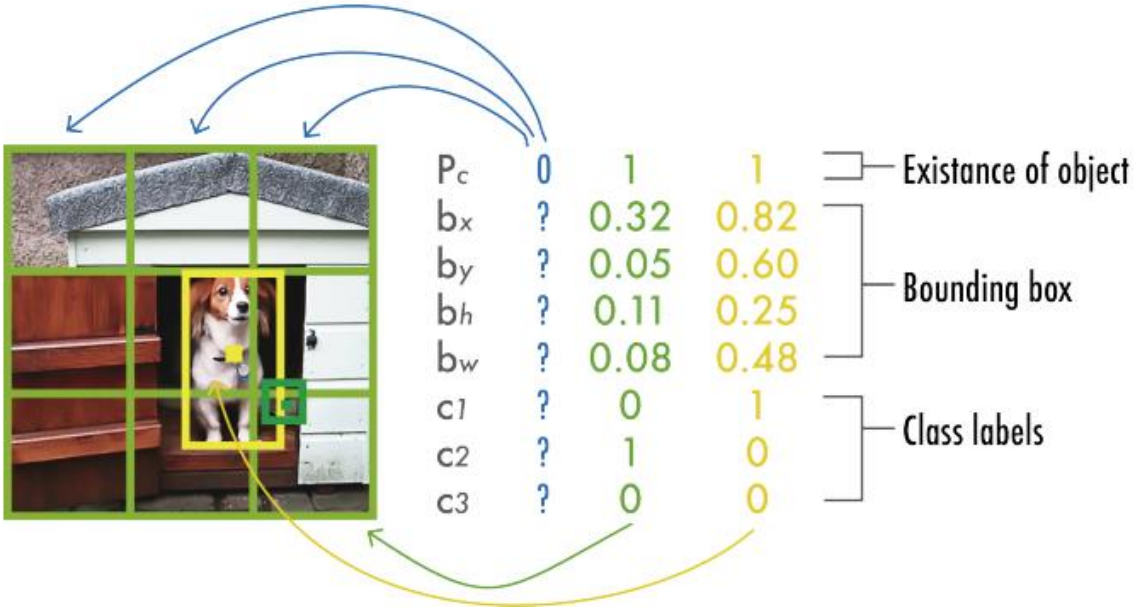

Figure 16 YOLO output prediction. The figure depicts a simplified YOLO model with a three-by-three grid, three classes, and a single class prediction per grid element to produce a vector of eight values.

2.9.3 YOLOv8

YOLOv8 utilizes a comparable underlying structure to YOLOv5, which is to be expected as both models were designed by the same company, Ultralitics. The C2f module, also known as the cross-stage partial bottleneck with two convolutions, enhances detection accuracy by integrating high-level features with contextual information. YOLOv8 employs an anchor-free model that utilizes a decoupled head to handle objectness, classification, and regression tasks separately. This design facilitates the concentration of each branch on its specific duty, hence enhancing the overall accuracy of the model. The sigmoid function was employed as the activation function for the objectness score in the output layer of YOLOv8. This function represents the chance that the bounding box includes an object. The model uses the softmax function to calculate the odds of each object belonging to different classes. YOLOv8 employs the CIoU [70] loss function for bounding-box loss, and binary cross-entropy for classification loss. These losses have enhanced the performance of object detection, particularly when handling tiny objects. The spatial pyramid pooling fast (SPPF) layer enhances computational speed by

consolidating information into a map of predetermined dimensions. Every convolution operation is accompanied by batch normalization and swiss activation.

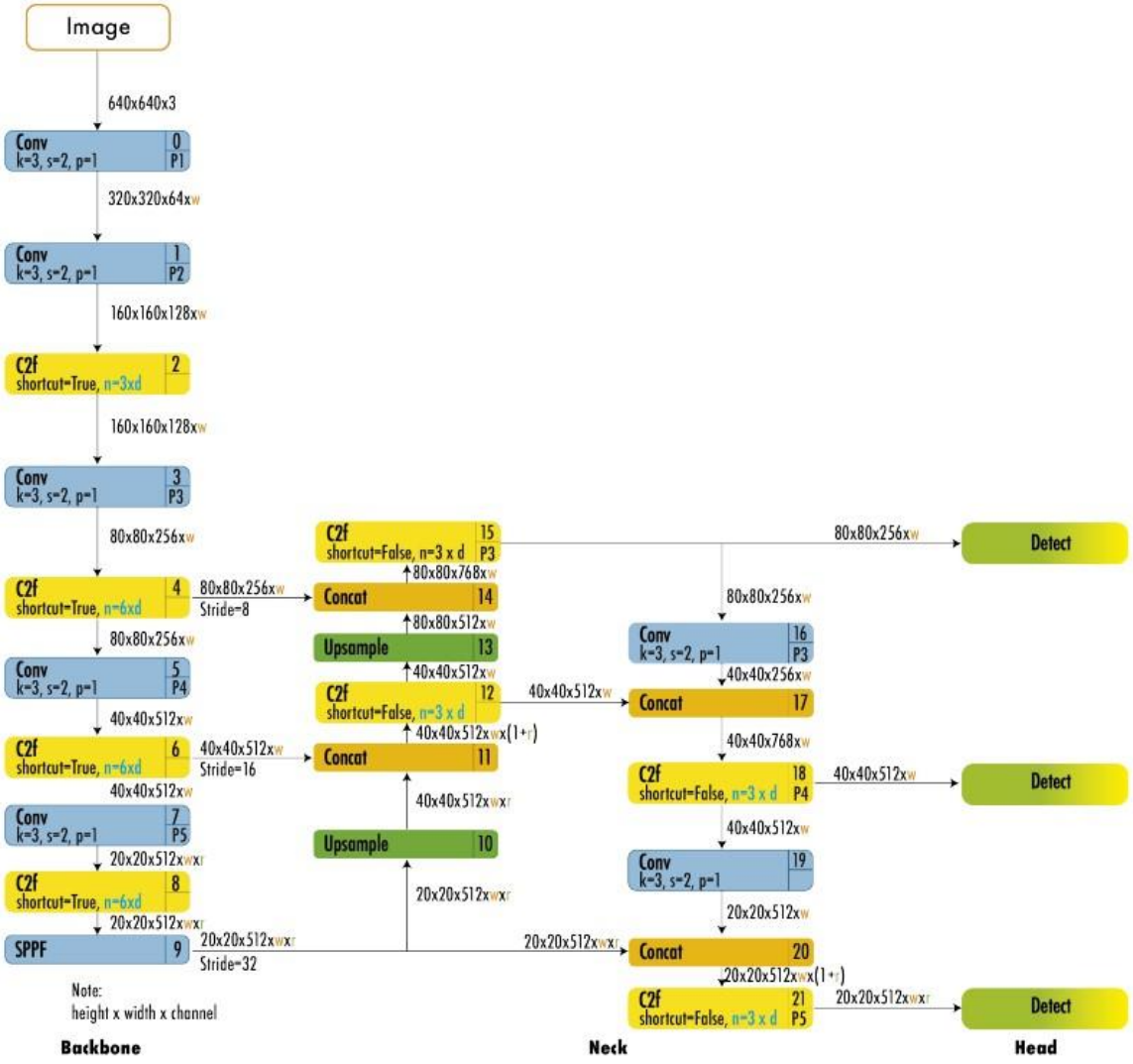

Figure 17 Image representation of the YOLOv8 model architecture

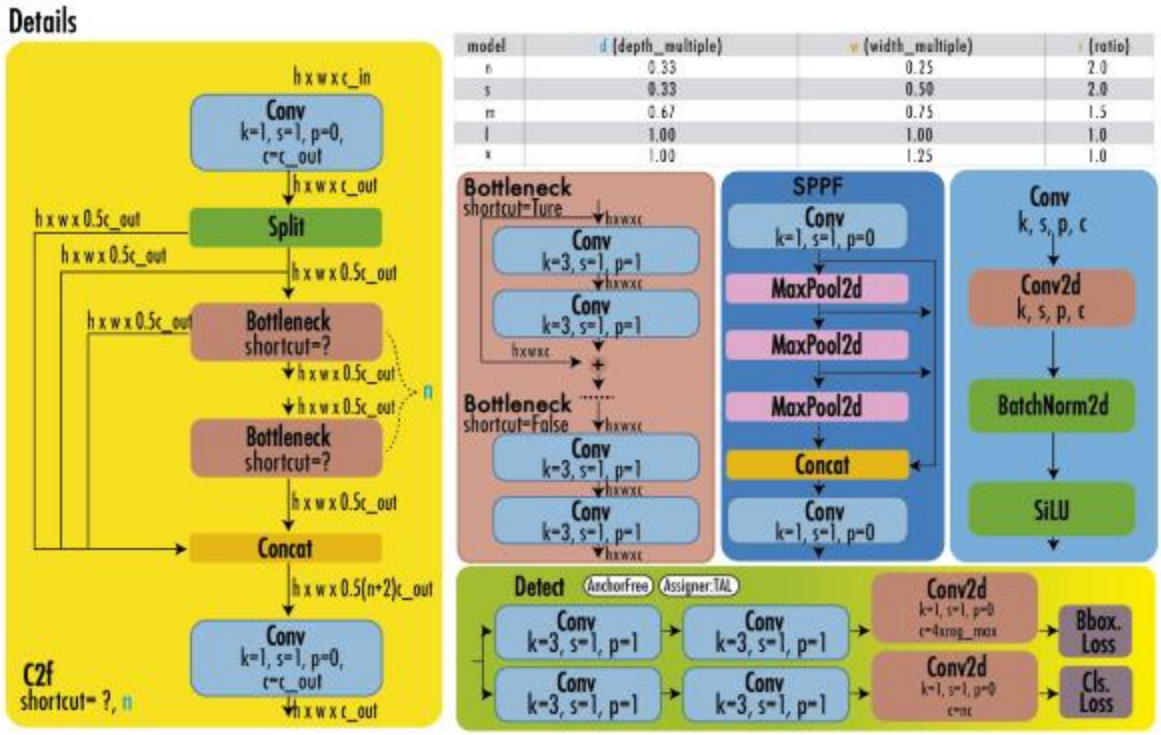

Figure 18 Details of the previously represented YOLOv8 model architecture

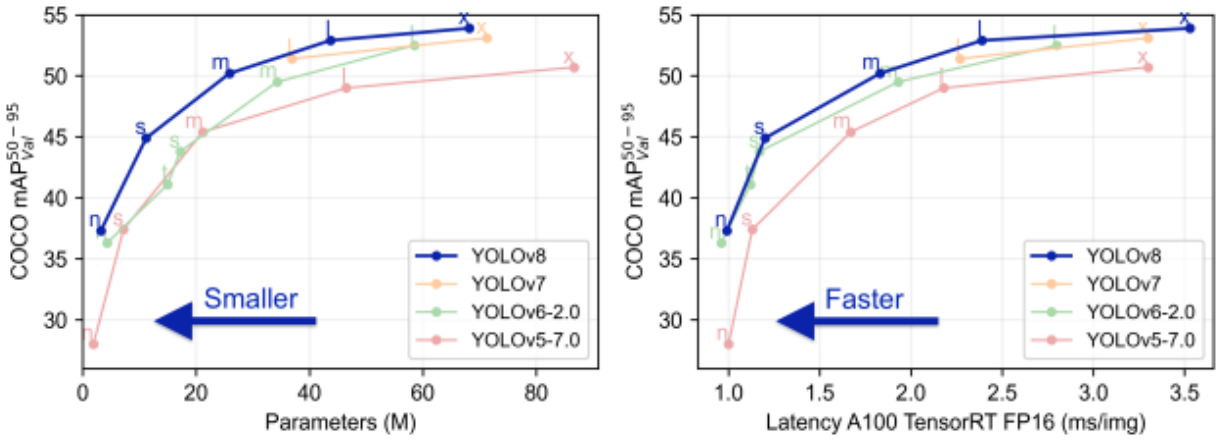

Figure 19 Performance comparison of YOLO object detection models. The left plot illustrates the relationship between model complexity (measured by the number of parameters) and detection accuracy (COCO mAP50-95). The right plot shows the tradeoff between inference speed and accuracy for the same models

2.9.4 YOLONAS

In May of 2023, Deci, a company that specializes in the development of production-grade models and tools for the purpose of constructing, optimizing, and deploying deep learning models, made YOLONAS [71][72] available to the public. As a result of its ability to detect small objects, improve localization accuracy, and enhance the performance-per-compute ratio, YOLONAS is considered to be suited for real-time applications. In addition to that, its open-source architecture is accessible for usage in research.

Among the many things that make YOLO-NAS unique are the following:

• Quantization-aware modules [73], also known as QSP and QCI, which combine re-parameterization for 8-bit quantization in order to minimize the accuracy loss that occurs during post-training quantization.

• Automatic architecture design using AutoNAC, Deci's NAS technology.

• A pre-training routine that includes data that has been automatically labeled.

The AutoNAC system, which played a crucial role in developing YOLO-NAS, is adaptable and capable of accommodating a range of factors such as data peculiarities and performance objectives. It helps users determine the optimal configuration that combines accuracy and processing speed for their specific needs. This technique takes into account the data, hardware, and other components involved in the process of drawing conclusions, including compilers and quantization.

They generated three architectures by varying the depth and positions of the QSP and QCI blocks: YOLO-NASS, YOLO-NASM, and YOLO-NASL (S, M, L for small, medium, and large, respectively).

Chapter 3 DATASET

Twenty thousand 24-by-40-pixel infrared pictures make up the dataset used in training our model [3]. In this study, 12 distinct classes of solar modules are provided, consisting of 11 classes representing various anomalies and 1 class representing No-Anomaly, where the panels are healthy. Each class is listed in Table 1 along with a description. This collection contains actual, distinct solar module anomalies. The Raptor Maps team used data collected from piloted aircraft and unmanned aerial systems with midwave or longwave infrared (3-13.5 μm) imaging systems to synthesize this dataset. Anomalies were divided into classes and cropped to the specific module. To improve accuracy, corresponding visible spectrum photos were employed for classification.

Class Name	Images
Cell	1,877
Cell-Multi	1,288
Cracking	941
Hot-Spot	251
Hot-Spot-Multi	247
Shadowing	1056
Diode	1,499
Diode-Multi	175
Vegetation	1,639
Soiling	205
Offline-Module	828
No-Anomaly	10,000

Figure 20 Composition of the images in our training dataset

A global report on airborne inspections discovered irregularities in 2.2% of all modules [73]. Comparison between the statistics reported in this dataset and the real-world regarding class proportions is important. In order for nominal conditions to be comprehensively identifiable, 10,000 No-Anomaly photos were included in the dataset. This could aid in locating anomalies that don't fall into any of the 11 classes that this specific dataset offers. Figure 2 illustrates the disparity in class composition by comparing the class percentages to the overall results. As there were more classes in the real world, the dataset proportions changed in size. The authors did their best to supply

as much verifiable data as they could so that researchers may readjust proportions to meet their requirements.

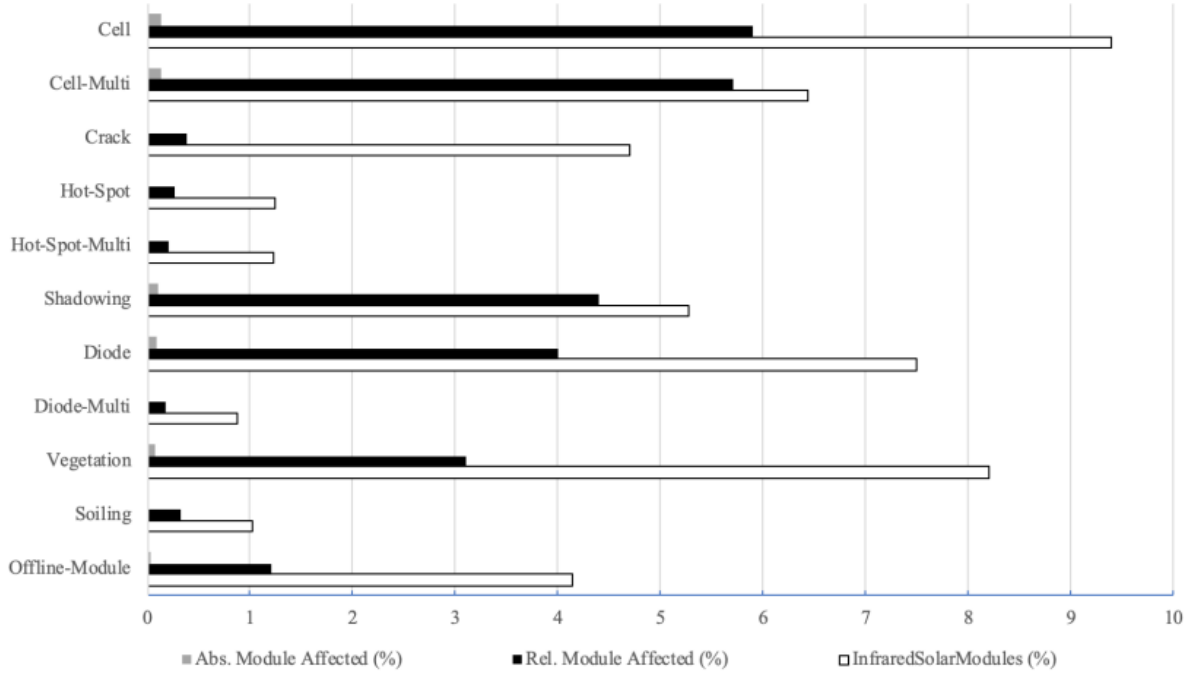

Figure 21 Graph depicting InfraredSolarModules dataset versus real world solar panel surface anomaly proportions

In this section we provide an explanation on what each type of anomaly refers to, as well as the power loss for each anomaly.

3.1 CELL

The cell anomaly class refers to one or more temperature-varying cells arranged in a rectangular shape. Shading or heavy soiling may be the cause of this fault on the surface of a photovoltaic panel. Losses will vary according to how many cells are impacted. Over time, elevated temperatures in both single and multiple cells can permanently degrade the cell and harm the bypass diode, resulting in a 33% reduction in power loss [17].

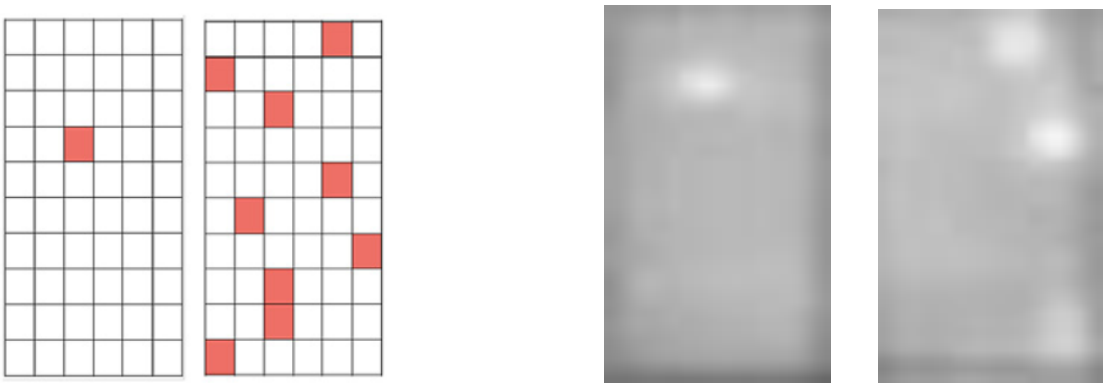

Figure 22 Visual representation of a single and multiple cell damage occurrences on a soral panel and some images from our database

3.2 DIODE

In order to shield PV modules from the shading impact and hotspot issue, bypass diodes serve as a bypass path for current flow [74]. A series-connected string of cells can be heated to reveal bypass diode damage. We refer to this type of surface fault as the diode anomaly class. When comparing a PV module with a bypass diode defect to one without, the power output of the former will be reduced by roughly 33% [75].

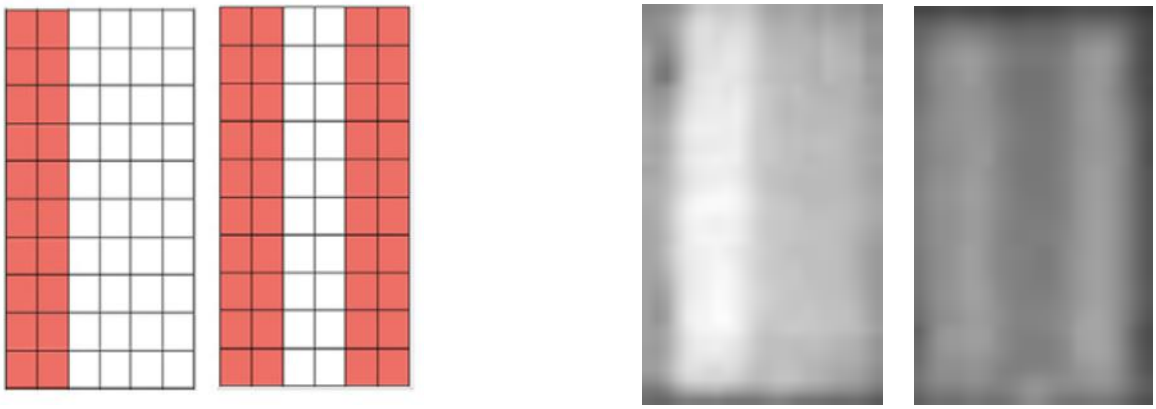

Figure 23 Visual representation of a single and multiple bypass diode damage on a soral panel and some images from our database

3.3 HOT SPOTS

We classify hot spots that appear as snail trails and discoloration on the surface of the affected solar panel as the hot spot and hot spot multi classes of anomalies. This is the only category that is diversified in its single and multiple counterparts. That decision was made because the phenotype of the fault differs when it comes to its multiple faults version. A 2% power loss can result from a snail trail hot spot [76]. Similar to the situation of single and multiple cells, losses are contingent upon the quantity of damaged cells. Over an extended period, elevated temperature cells may permanently degrade the cell and harm the bypass diode, resulting in a 33% reduction in power loss.

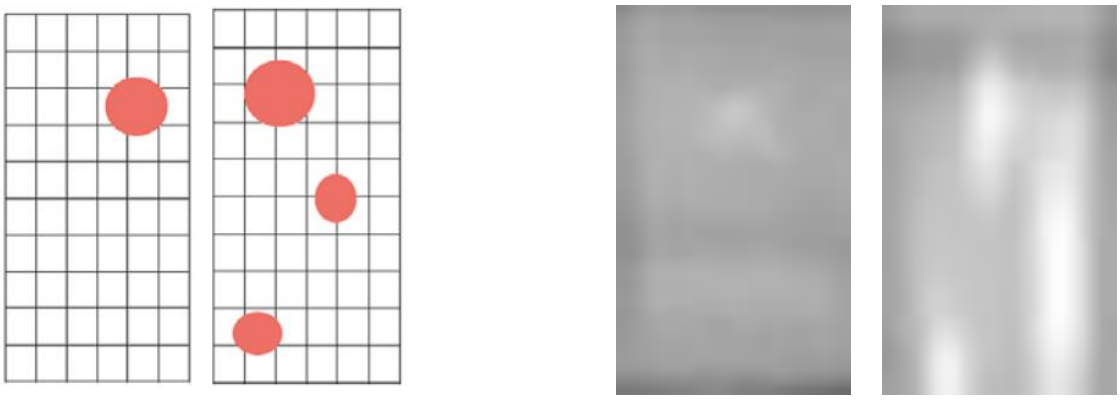

Figure 24 Visual representation of a single and multiple hot spots on a soral panel and some images from our database

3.4 SHADOWING

The shadowing class of anomalies is comprised of images from when the photovoltaic array's shaded modules get less radiation during partial shading than its unshaded modules [77]. A PV module's maximum power may decline dramatically in a shaded environment, lowering the system's energy yield [78]. The power output of the solar module will be reduced by 20–40% when shaded [79, 80].

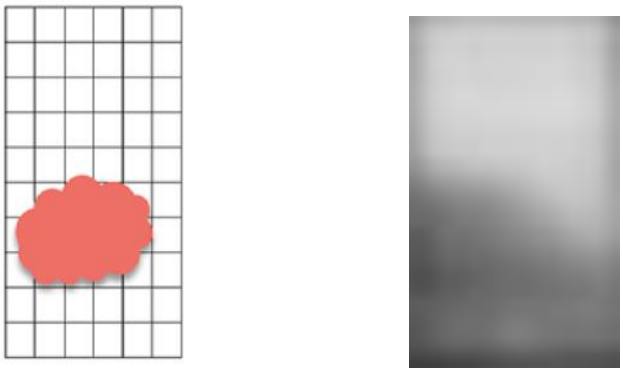

Figure 25 Visual representation of shadowing on a soral panel and an image from our database

3.5 OFFLINE MODULE

The offline module class of faults contains images from modules that have been disconnected from the PV system. This can happen to a string of solar panels or a singular one. The affected module does not have power output.

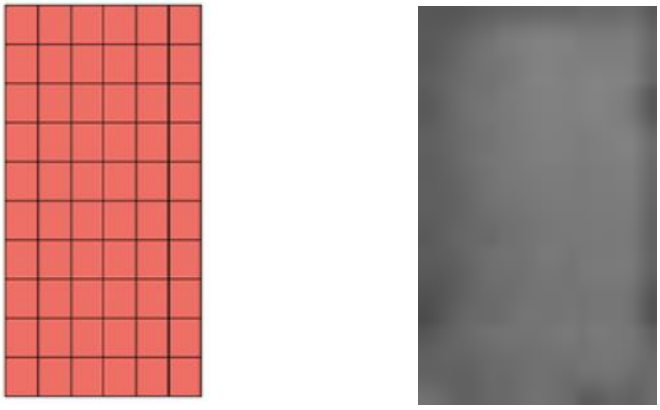

Figure 26 Visual representation of an offline module and an image from our database

3.6 SOILING

Soiling losses refer to power loss resulting from snow, dirt, dust, and other particles that cover the surface of the PV module . We can expect power loss of 1.5%-6.2% depending on the location of the PV plant [81].

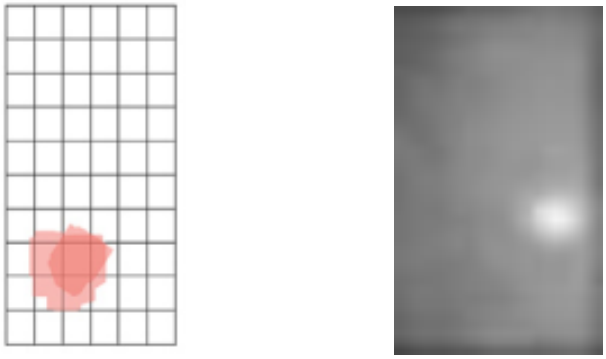

Figure 27 Visual representation of soiling on a soral panel and an image from our database

Chapter 4 METRICS

After each time we modified either the parameters of our model, the data that was used to train the model or even when we changed the model all together, we had to assess how well the training went and how capable the model produced by the training was in detecting anomalies photovoltaic modules. This process is called the evaluation process, and it is essential when it comes to neural network training, since it provides us with statistics what can be used to compare multiple trainings and therefore, choose the most efficient model for our particular case. The comparison process requires the use of certain metrics in order to thoroughly assess the evaluation power of the generated models and compare them on an equal footing. It is important to mention that the validation mode of the YOLOv8 model produces the graphs of these metrics at the end of the training process. As a result, some of the metrics we employ in our work are as follows:

4.1 THE CONFUSION MATRIX

Four outputs come from a binary classification, and these can be summed up in a confusion matrix. The values from the matrix are then used to generate classification metrics. These four possible results are called true positive, false positive, false negative and true negative. We are dealing with a true positive case when the model correctly predicted the input being true when the actual observation was also true, meaning with our model correctly identified an anomaly on the surface of a solar panel. A false positive result refers to a prediction that is true when the input was false, meaning of the model found an anomaly that was not present on a panel and thus triggering a false alarm. A false negative case is when the model made a prediction that was false when the observation was really true, meaning the model didn't accurately pinpoint an error that was present, resulting to a defective panel, which will lead to it getting no further attention. Lastly true negative means that the model's prediction was false and the observation itself was also false, meaning that the model accurately identified the lack of any anomaly on the surface of the panel. The final result of the YOLOv8 model training results in the production of two PNG pictures, one of them being the confusion Matrix and the other one being the normalized confusion, Matrix. The first one gives us the actual numbers of each of the four cases, while the normalized version gives us the percentage of each of the four cases, making it simpler to compare the performance across classes. It is also important to understand that when we are dealing with more than two cases, meaning that the decision is not binary, the predicted line in the actual

column creates multiple intersecting blocks that indicate the effectiveness of our model. Then we can observe whether the model correctly identifies a fault, if it identifies a fault but classifies it as another class of faults, or if it does not identify the fault altogether. In this thesis we have made also use of normalized confusion matrixes with multiple columns, because our final trained model can distinguish and classify multiple photovoltaic module anomalies. Some other very important metrics to the assessment of our model's effectiveness can be calculated from the results that we get from the confusion Matrix.

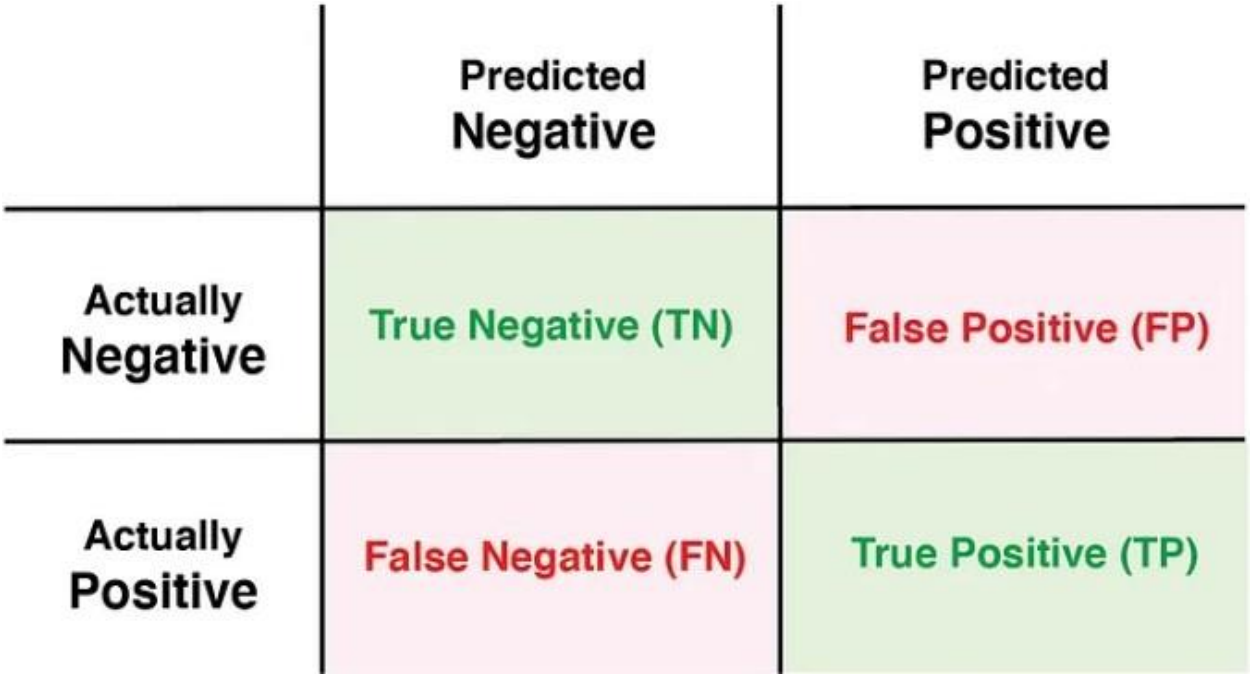

Figure 28 Visual representation of a confusion matrix

4.2 ACCURACY

The most fundamental indication metric is accuracy, which indicates what percentage of all predictions our model correctly predicted. Accuracy fails for imbalanced datasets even though it is simple to compute and comprehend. In this thesis, accuracy will not be included in the comparison process.

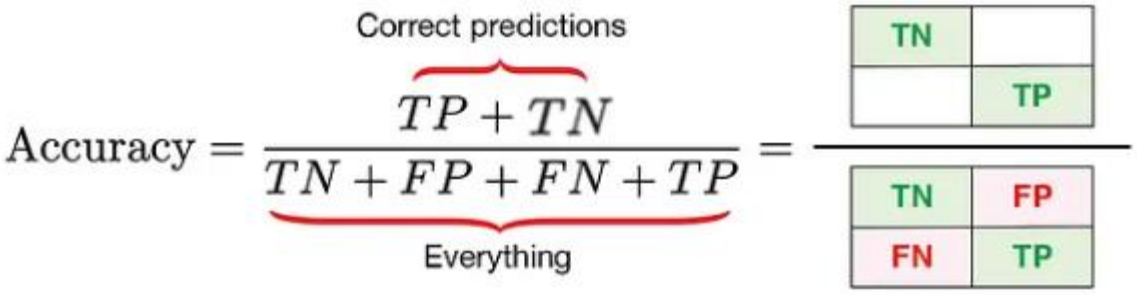

Figure 29 Mathematical and visual representation of the accuracy metric

4.3 PRECISION

The fraction of correct positive predictions is provided by precision. Minimizing false positive errors is given priority in this metric. YOLOv8's Validation mode generates a precision curve, meaning an image showing precision values at various cutoff points. This curve makes it easier to see how precision changes with threshold.

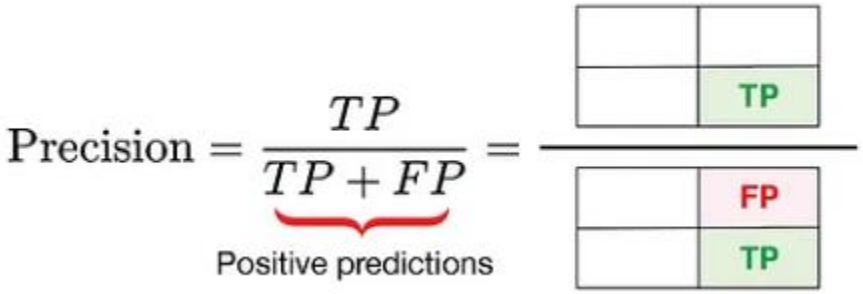

Figure 30 Mathematical and visual representation of the precision metric

4.4 RECALL

Recall measures the model's capacity to identify every instance of a class by calculating the fraction of true positives among all actual positives. It places a higher priority on minimizing false negative mistakes. YOLOv8's Validation mode generates a recall curve, meaning an image showing recall values at various cutoff points. This curve becomes especially significant when dealing with imbalanced classes.

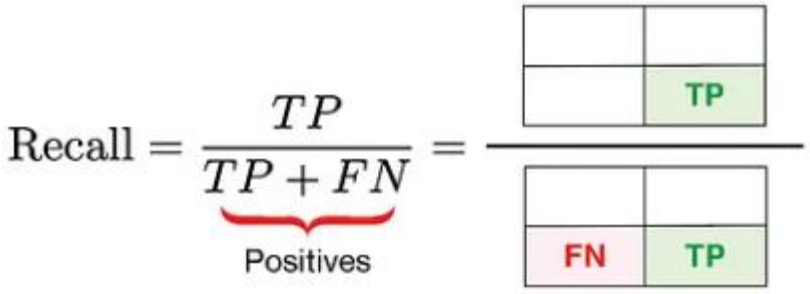

Figure 31 Mathematical and visual representation of the recall metric

4.5 F1 SCORE

The F1-score, which is resilient against imbalanced datasets and combines precision and recall, is a sophisticated statistic that offers the best of both worlds. The metric can be computed directly from the confusion matrix, or it can be defined as the harmonic mean of precision and recall. YOLOv8's Validation mode generates a recall curve, meaning an image showing F1 values at various cutoff points. Interpreting this curve can offer insights into the model's balance between false positives and false negatives over different thresholds.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

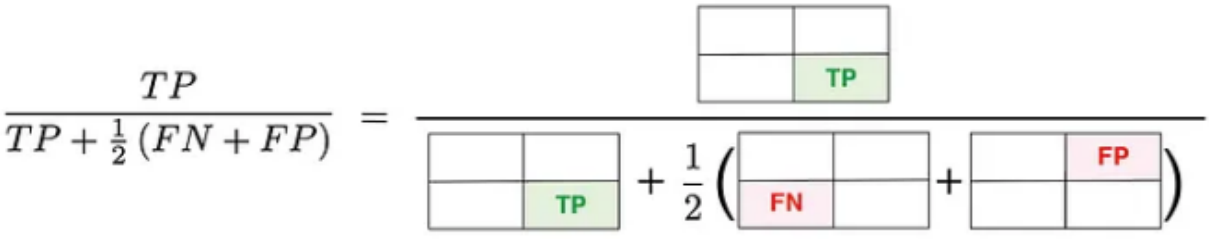

Figure 32 Mathematical and visual representation of the F1 score metric

4.6 INTERSECTION OVER UNION

The metric called intersection over union (IoU) is the ratio of the intersection area to the union area of the predicted bounding box and the ground truth bounding box. The predicted bounding box is generated by our trained model, whereas the ground truth bounding box is the one we drew in the annotation part of our experimentation process. We will expand on the annotation process in chapter 5 of our thesis. IoU measures the overlap between the ground truth and predicted bounding boxes and it is essential for the accurate assessment of object localization.

4.7 MEAN AVERAGE PERCISION, mAP50 & mAP50-95

The precision and recall performance of the model are combined into another single metric called Average Precision (AP), which calculates the area under the precision-recall curve. By computing the average AP values over several object types, Mean Average Precision (mAP) expands on the idea of AP. This metric is helpful to give a thorough assessment of the model's performance in multi-class object detection settings. Mean average precision (mAP50) is the value obtained at an intersection over union (IoU) threshold of 0.50. It is an indicator of how accurate the model is when simply taking into account "easy" detections. When we address the metric called mAP50-95, we refer to the mean average precision averaged over a range of IoU thresholds, from 0.50 to 0.95. It provides a thorough understanding of the model's performance at various detection difficulty levels.

Chapter 5 EXPERIMENTATION PROCESS

5.1 HARDWARE THAT WAS USED

In order to prepare for further analysis of the procedures that we carried out during the process of experimentation and model training, we are going to make a brief note of the instruments that were utilized throughout the entirety of the process. All of the model training cycles were executed on the Google Colab environment, which served as the primary platform for their development. Python served as the sole programming language for the development of the various scripts, including those used for the training of the models. Any miscellaneous python script was executed directly via the use of our computer's command prompt. The personal computer on which the code was written does not possess the intense processing power that is required for a model of our scale to be trained. This is the case when it comes to its processing hardware. The utilization of Google Colab's Pro cloud services was the solution that we ultimately put into action. For a small fee, we were granted access to Google's remote and powerful graphics processing unit (GPUs). The Tesla T4 GPU was slower but consumed fewer process units, it was primarily utilized during the night for conducting longer training sessions. In order to complete multiple training courses throughout the day, the V100 and A100 GPUs were utilized because they were the two options that offered the fastest performance. In conclusion, it is important to mention that we had access to 13 gigabytes of random-access memory (RAM) through the cloud services.

5.2 SORTING PROCESS

It is worth mentioning that the initial collection of our infrared solar panel dataset did not have any sorting applied to it. The dataset contained a collection of images organized in a folder, along with a JSON file that detailed the anomalies found in each image. We utilized a Python script to sort our data. The script efficiently navigated through the JSON file and image folder, extracting the necessary information from the JSON file, and relocating the corresponding image to a newly created folder named after the specific anomaly. Nine folders were created to store images of healthy solar panels without any anomalies. It became clear that these images required additional sorting before they could be effectively used to train the YOLO models. The images were divided into different folders based on their categories. The majority, 70% of the

total images, were placed in the training folder. Another 20% of the images were allocated to the validation folder, while the remaining 10% were assigned to the testing folder. This was the ratio we used for the training of our model.

5.3 ANNOTATION PROCESS

The next step we undertook in the training of our model was called data annotation. In order to understand the need for this step, we shall first explain the training requirements for the YOLO line of models. For this type of model to be trained the data that we need to feed them must adhere to a specific structure. Each image that shall be used for the training or validation processes shall be accompanied by a text file, called a label. The text file must have the same name as the image in question. This text file contains the coordinates of the bounding box that indicate the failure or the anomaly within the image, as well as the class of the anomaly. Data annotation is the process of manually creating these text files, or labels as they are referred to, in order for our data to be complete and for it to have the appropriate structure to be used for training and validation. Writing all these text files by hand is a mere impossible task, and one not worth pursuing. Lucky for us, there are plenty of free tools online that can help us with this process. The tool we used is a website called CVAT, short for computer vision annotation tool. This application allows us to import all the images of the data set and through the use of a graphic user interface, gives us the ability to draw the rectangle, the bounding box, is that indicates where the anomaly is present on each solar panel. Annotating almost 10,000 images using this method still took considerable amount of time. After the completion of this process this site exports all this information by creating the appropriate labels named exactly as the picture that was used to create them, containing the coordinates of the bounding boxes we created. All left to do was to create two folders, one named “images” and the other named “labels” and put the specific data that we wanted to feed our model for its training cycle. Because we were unsure of the performance of the YOLO models, we experimented with multiple trainings, each time containing more annotated images, until we finally annotated almost the whole data set. More details about the size and the reasoning behind our multiple training cycles shall be further explained later in the current chapter of this thesis.

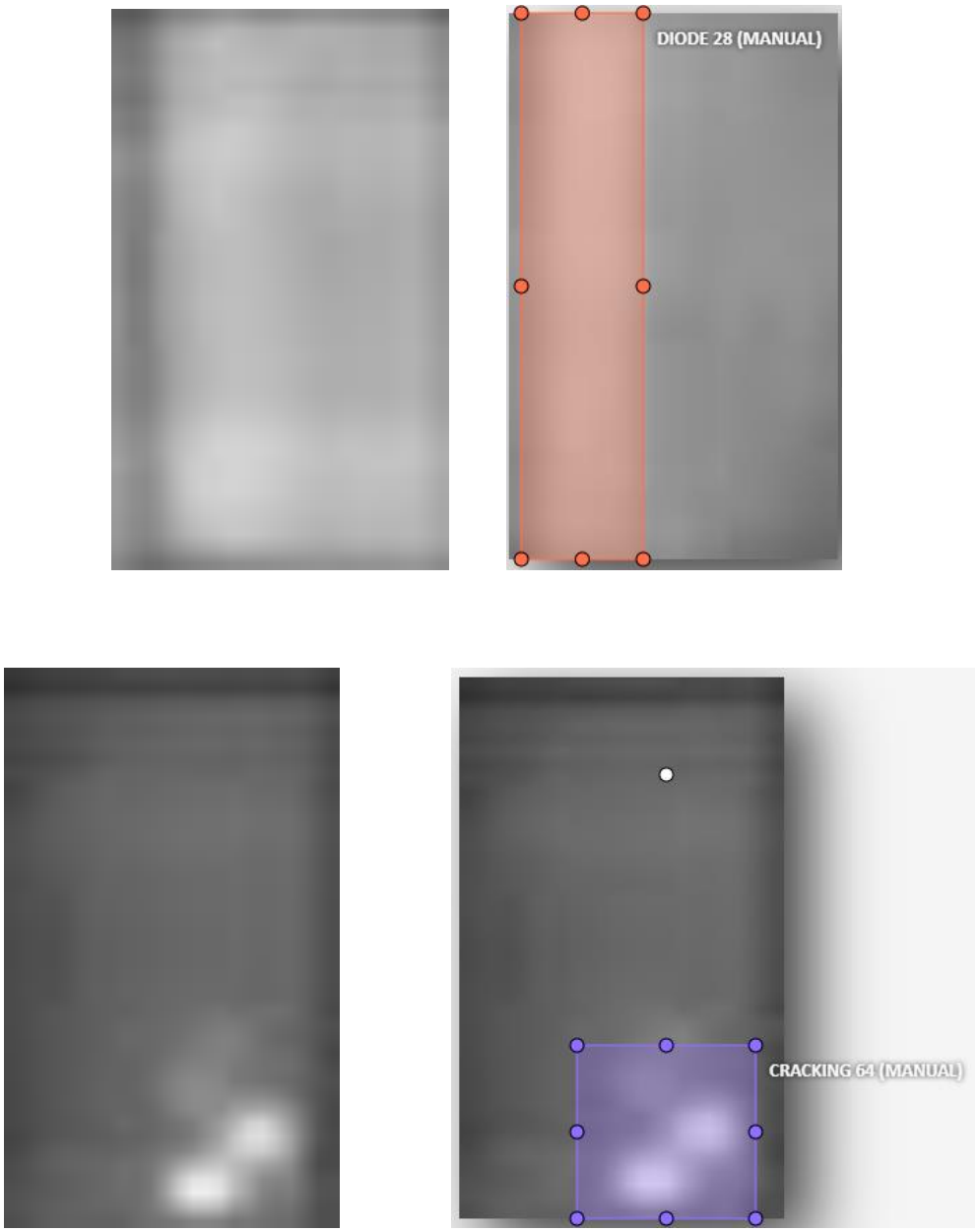

Figure 33 Some examples of the manual annotation progress

5.4 FIRST TRAINING

When we first began the process of training our model, we had not yet experimented with any other models or annotated data. As a result, we only utilized a small portion of the total data and included fewer categories than the final training data set. Images of

panels that contained the cell, multiple cell, diode, multiple diode, cracking, and offline module classes of anomalies, in order to determine whether or not our model is effective on a more limited data set. Additionally, the number of training epochs was the first hyperparameter that we made adjustments on. The YOLOv8n (nano) model was trained using this smaller data set for a total of 20, 50, 70, 80, 100, 120 epochs. The YOLOv8l (large) model was also used to carry out one training session that lasted for 20 epochs. Within the context of these preliminary trainings, no additional hyperparameters were modified. The time needed for a single epoch to be trained on the nano model was on average 32 seconds, while the large model needed an average of 47 seconds.

5.5 INCLUSIVE DATASET

Continuing with the annotation of more images is the next logical step in order to enhance inclusivity in surface anomaly categories. This time, we annotated images to ensure that our model can detect anomalies related to shadowing, vegetation, and soiling. We train our model twice using this data set, once for 80 epochs and once for 100 epochs. After careful analysis and evaluation, it became evident that making modifications to this specific data set was necessary to improve the performance of our model. The details of this conclusion will be further discussed in chapter 6 of this thesis. This change involved eliminating the category of anomalies related to soiling. Within our dataset, we observed small images measuring 24 by 40 pixels. Interestingly, the soiling anomaly exhibited a striking resemblance to the cell anomaly in terms of appearance. The cell anomaly comprised 1877 images, the multiple cell anomaly comprised 1288 images, totaling 3165 images, which accounts for approximately 15.825% of the total images in our dataset. In contrast, the soiling category consisted of only 205 images, accounting for a mere 1.025% of the total images in our dataset. Due to the uneven distribution of cell class images and soiling class images, our model mistakenly identified images with soiling errors as images with cell errors. When it comes to the anomaly on the panel's surface, we found that these two categories are remarkably similar in nature, cause [17], and appearance when viewed through the infrared spectrum. As a result, we made the decision to rebrand the images that depict the soiling faults as cell faults. During the analysis of this modified dataset, we conducted four training sessions for the YOLOv8n model, each lasting for different

numbers of epochs: 80, 85, 90, and 100. The training and validation process for each epoch took on average 43 seconds.

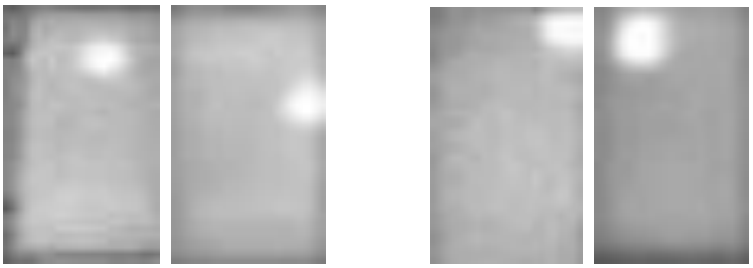

Figure 34 On the left, there are 2 random images of solar panels containing the cell class anomaly. On the left, there are 2 random images of solar panels containing the cell class anomaly. We depict those 4 images to highlight the similarity between these 2 surface anomalies on the infrared spectrum

5.6 COMPLETE DATASET

As we proceeded, we annotated the last of the remaining images of faulty solar panels and included the final two categories, which were the hotspot category and the multiple hotspot category. Following the conclusion of the annotation process, the YOLOv8 model was provided with this complete data set. In this particular instance, we sought to achieve the effective detection and identification of a wide variety of surface faults by experimenting with the modification of multiple hyperparameters. Our goal was to create the ideal conditions for accomplishing this.

The hyperparameters that we altered were the same ones that we examined in chapter 2 of this thesis. These hyperparameters included the training epoch count, the batch size, the optimizer, and the learning rate. There was a total of sixteen new training courses that were finished, and all of the results were compared with each other and analyzed in order to determine which set of hyperparameters was the most optimal. In chapter 6 of this thesis, we will discuss the outcomes of the validation process that was performed on these trainings. The training and validation process for each epoch took on average 49 seconds.

5.7 YOLONAS MODEL

Given our data was annotated in the YOLO format, we decided to explore training a different model in addition to YOLOv8. Our choice fell on YOLONAS, an acronym for "you only look once neural architecture search", seeing how it excels at small object detection. Working with this model was incredibly challenging due to the extensive computing time and cloud resources required. The YOLOv8 model efficiently generated cache files for both the images and labels in the dataset. These files were generated when the data set was initially inputted into the model, allowing for rapid loading of the dataset and the start of training. YOLONAS does not generate any files of that type, so each time we initiate a new training cycle, the data needed to be freshly loaded. Our dataset required approximately 48 minutes on the fastest GPU available to us, the A100 GPU, and 95 minutes on a tesla T4 GPU each time in order to be loaded. Training YOLONAS posed another significant issue as the training process would abruptly halt, resulting in wasted processing units and valuable time. The error message indicated that an image was not found in the data set, despite its clear presence. In spite of these limitations, we were able to train this model successfully on two occasions, one for 100 and another for 120 epochs. The training and validation process for each epoch took on average 9 minutes.

5.8 BINARY DETECTION MODEL

After conducting all the multi class identification experiments mentioned above, we delved into the realm of binary detection training using YOLOv8. This involved training the model to determine whether a fault exists on the surface of a panel or if it is completely healthy. In order to tackle this, we devised a strategy involving the creation of two distinct classes: healthy and faulty. The annotation process had a unique approach compared to previous trainings, as manually annotating 20,000 images would be quite laborious. The bounding box we needed to draw on its image remained consistent, encompassing the entire image. A Python script was developed that efficiently parsed both the faulty and healthy images, generating a text file with the same line of code for each image, its label. Every text file, or label, includes the parameter for either the faulty or healthy class, depending on whether an anomaly exists

in the associated image. It also contains the dimensions of the bounding box, which are uniform because it is a complete image. This method generated the necessary labels for the completion of the YOLO format quickly. Consequently, the dataset was prepared to be fed into our model once more. We conducted the first training session using YOLOv8n (nano) for 100 epochs. Based on the initial training results, we proceeded with training the session for an 80, 60, 50, 40, 30 and 20 epochs. The decrease of the epoch number was applied, so that the point of maximum return could be found. After the epoch count altering, we experimented by changing the batch size and learning rate hyperparameters of the model. Training each epoch required approximately 1 minute and 40 seconds.

5.9 TESTING PROCESS

Upon completing the training and validation stages of our experimentation with models mentioned above, it has become evident that testing was necessary. We carefully selected the top models based on validation and thoroughly tested their performance using a variety of images. There are some images that came from the same data set we used for training and validating. These images make up the last 10% that we did not use in the other two processes. However, to enhance the reliability of the testing process, several images from different data sets were utilized in this phase. Locating these images necessitated scouring the internet, yet the images we found did not adhere to the same format as the one utilized for our training. We had to modify our new inputs to ensure they could be understood by our models. We developed a Python script that resized the images to a uniform dimension of 24x40 pixels and converted them to grayscale. Another issue that arose was the limited number of images in this new collection. However, we employed a technique known as data augmentation to address this. By utilizing functions like flip and rotate, we were able to generate additional images, ensuring a smoother testing process. The findings from our testing will be elaborated on in subsequent sections of the thesis.

Figure 35 One of the initial images that we sourced from the web. The panel in this image has easily recognizable damage spots.

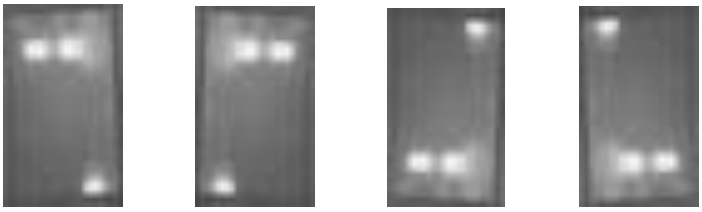

Figure 36 The 4 images that we generated using the data augmentation functions flip and rotate. These images were created after the downscaling and grayscaling of the original image.

Chapter 6 RESULTS AND COMPARISONS

In this section, the results of our experimentation process will be compared to each other, in order to assess the effectiveness of our training sessions and the robustness of the resulting models. Multiple metrics shall be used, the model size, the epochs count, the optimizer, the batch count and the learning rate of the models, in order for us to draw a conclusion and determine which one of the training sessions resulted in the most well optimized model. Both the binary classification and multi class classification types of models shall be examined, since we search for the best one from both types. Throughout all the tables presented below, the validation metrics that are used for the comparisons are the F1 Score and the mAP50-90 score since, as we explained in chapter 3, these are the most sophisticated metrics. All the results from these training cycles are also presented in table form ate the end of this thesis, in chapter 9, the appendix.

6.1 EPOCH COUNT ON THE SMALL DATASET

Firstly, we compare the results from the very first training and validating sessions of the model YOLOv8 nano. This was our first experience in CNN training in general. Images from the Cell, Diode, Cracking and Offline Module classes of anomalies were included in the dataset used for these initial training cycles. These training sessions were conducted, so that we could assess the effectiveness of the YOLOv8 model on our annotated dataset, they were never intended to be our finalized model.

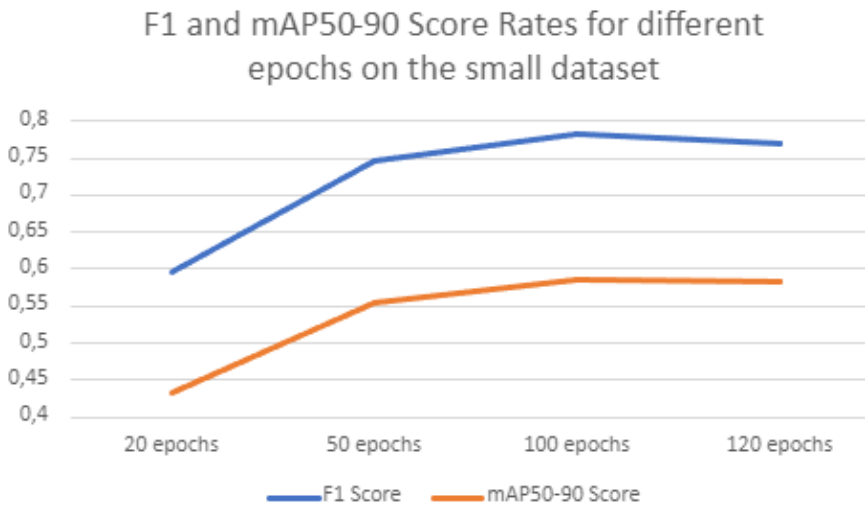

Figure 37 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the small dataset for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01

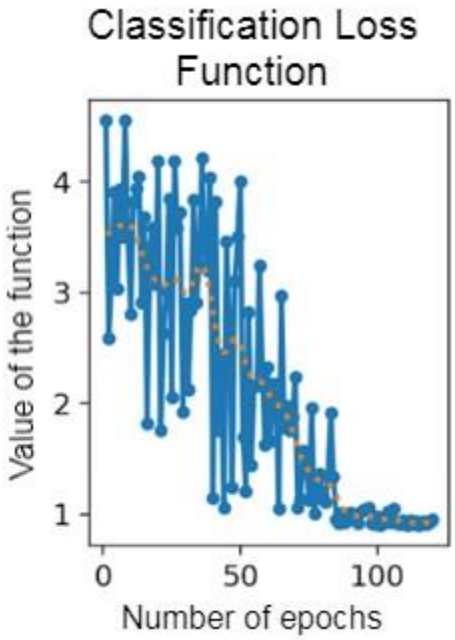

Figure 38 Classification loss function graph for the training of the YOLOv8 nano model for 120 epochs on the small dataset

From our first batch of training sessions, aside from the effectiveness of the YOLOv8 model on the annotated data from our database, we tested the effects that the number of epochs have on the training process of this model. The conclusion that was reached

was that the ideal number of epochs trained is close to 100, for the multi-class classification model. When we train the model for fewer epochs, the metrics signify that the model can be improved, and when the model is trained for more epochs, we reach the point of stagnation. Then, both the loss functions, as seen in figure 39 above, and the metrics mentioned above, in figure 38, do not improve, indicating that further training is useless.

6.2 FURTHER ANOMALY CLASSES INCLUSION

In these 2 training sessions we included 3 new classes of anomalies: vegetation, shadowing, and soiling. No further modifications were made to the previous model. We were testing the model's reaction to the introduction of more annotated images. The results are presented below.

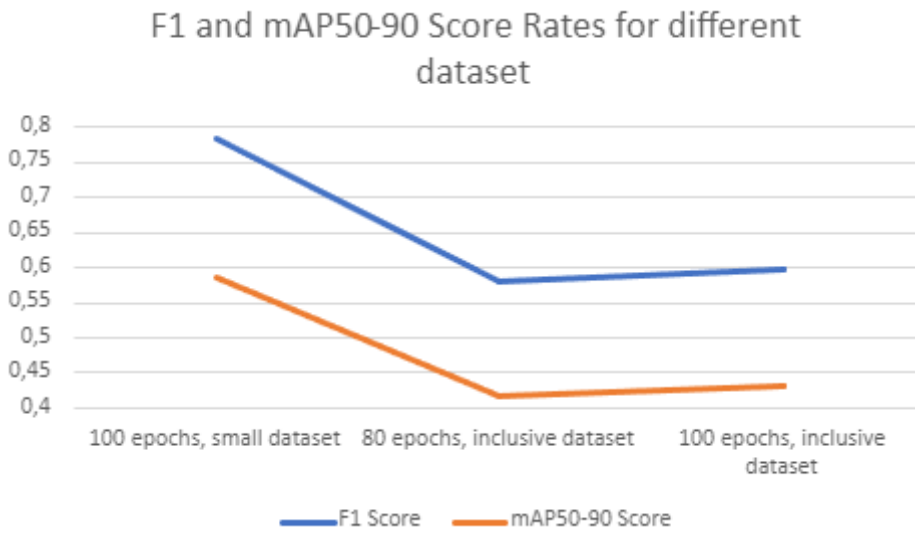

Figure 39 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the small and more inclusive datasets for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01

After this direct comparison, we can clearly detect that the metrics of the model decrease with the inclusion of more anomaly classes. More specifically, we see an 18,5%

decrease in the F1 score and a 15.5% decrease in the mAP50-90 score. After reviewing the low true detection rate of the soiling class (10% for the training session for 80 epochs and 14% for the training session for 100 epochs) we decided that modifying the dataset would be beneficial for future training sessions. The modification in this case, as mentioned in the previous chapter of this thesis, is the removal of the Soiling class, and the reannotation of the images in this class to the Cell class instead.

6.3 POST DATASET MODIFICATION

After the above-mentioned modification, we trained the YOLOv8 model twice, again for 80 and 100 epochs, in order to see if this reannotation of the soiling class images would improve the overall performance of our model.

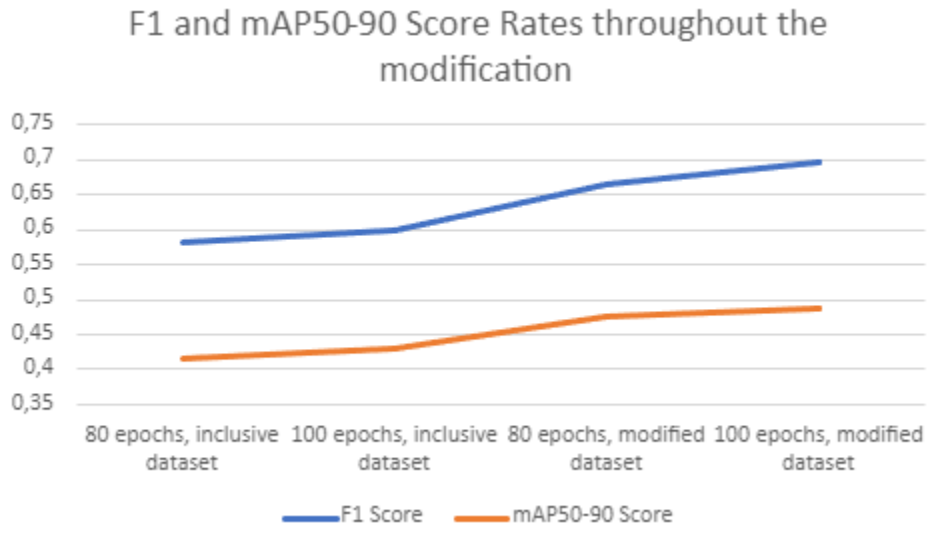

Figure 40 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the more inclusive and modified datasets for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01

We can access from the true detection rates of the other classes that the modification of the dataset was a correct course of action. The validation metrics from these trainings also provide reassurance for our modification, since all the values for all these metrics

increased significantly after the reannotation of the Soiling class. More specifically, the 2 most decisive validation metrics, the F1 score and the mAP50-90, were increased by 9.8% and 5.8%, respectively.

6.4 MULTIPLE HYPERPARAMETER COMPARISON ON THE COMPLETE DATASET

After our modification that was mentioned above, we opted to include the final 2 classes of anomalies, being Hot Spot and Hot Spot Multi. After the inclusion of all the classes, we started experimenting with all the hyperparameters that were mentioned in chapter 2 of this thesis, those being the Optimizer, the Batch Size, and the Learning Rate. Our goal was to find the best possible combination, in order to achieve the best detection rates and metrics possible for the YOLOv8 model and this dataset.

6.4.1 OPTIMIZER COMPARISON

Table 2 Comparison matrix between some of the different optimizers that were available to us. The YOLOv8 model has a standard list of modifiers that can be applied to it.

Metrics/ Models	YOLOv8 small, 100 epochs, batch size = 16, learning rate=0.01			
	AdamW Optimizer	Adam Optimizer	SGD Optimizer	RAdam Optimizer
F1	**0.62**	0.59	0.60	0.56
MAP50-90	**0.40**	0.40	0.39	0.37

As we can assess, the AdamW optimizer, which is the default one that the YOLOv8 will use if we do not specifically modify this hyperparameter, offers us the best metrics out of the other options in this list, while keeping all the other hyperparameters unchanged. This made us proceed with using the AdamW optimizer for the rest of the training cycles that were conducted.

6.4.2 BATCH SIZE COMPARISON

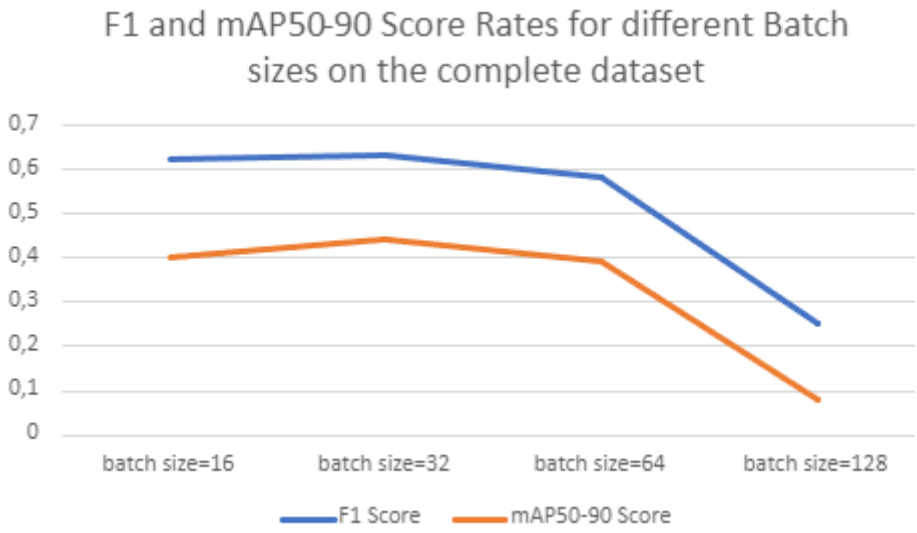

Figure 41 Line graph of the F1 and mAP50-90 scores Rates for different batch sizes on the complete dataset for the YOLOv8 small model. The model's number of epochs is 100 and the model's learning rate is 0.01

As we mentioned in chapter 2, it is optimal to assign the batch size to powers of 2, in order to allow the GPU to most efficiently make use of its memory. From this comparison matrix, the conclusion that can be made is that increasing the batch size to 32 improves both decisive metrics of the model specifically the mAP50-90, where we see an increase of 3.5%. The 0.9% increase in the F1 score may not be so noteworthy. Further increase seems to not be beneficial since we can see a decrease of 4.1% in the F1 score and a 0.1% decrease in mAP50-90. The validation metrics plummeted when the batch size was increased to 128, with a decrease of 37% and 32% in the F1 score and mAP50-90, respectively.

6.4.3 LEARNING RATE COMPARISON

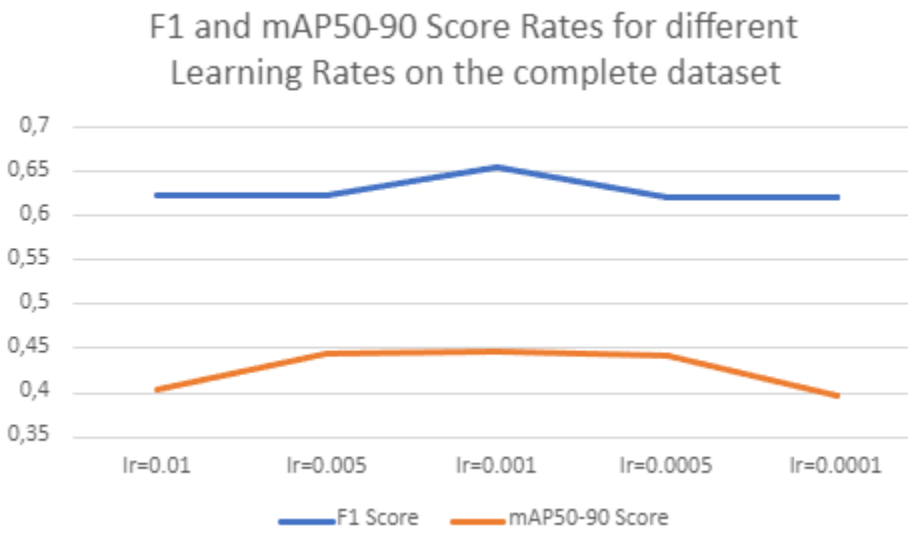

Figure 42 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the complete dataset for the YOLOv8 small model. The model's number of epochs is 100 and the model's batch size is 16

The range between 0.01 and 0.0001 is the most widely used range of learning rates when it comes to machine learning. It is apparent from the matrix presented above that setting the learning rate of our model to 0.001 increases our validation metrics. The F1 score improves by 3.2% and the mAP50-90 improves by 4%. Meanwhile setting the learning rate to either 0.0001, 0.005 or 0.0005, which are commonly used learning rates, does not seem to either benefit or hinder the model's performance.

6.4.4 FINAL COMPARISONS FOR THE MULTI- CLASS CLASSIFICATION MODEL

In this part of our experimentation, as mentioned in chapter 5, we combined various hyperparameters in order to assess which combination would wield the best results, in terms of metrics. All the previous training cycles and their validation statistics were taken into consideration, since the combination of all our conclusions creates indeed the most robust network yet. If we combine the most efficient optimizer, batch size and learning

rate, in this case he AdamW optimizer, batch size equals to 32 and learning rate equals to 0.001, we create a model, whose validation metrics are the best we have encountered for an 8-class classification model. It is also important to mention that when we turned the batch to 128, the memory that was available to us through the cloud service did not suffice for the training of the model when the learning rate was 0.001 or 0.0001 and the training sessions were terminated instantly, hence the lack of metrics for those cycles. If we had the available resources, the metrics from those training would most likely be exceptionally low, as seen in the training session with the hyperparameter combination of 100 epochs, AdamW optimizer, learning rate equal to 0.01 and batch size equal to 128.

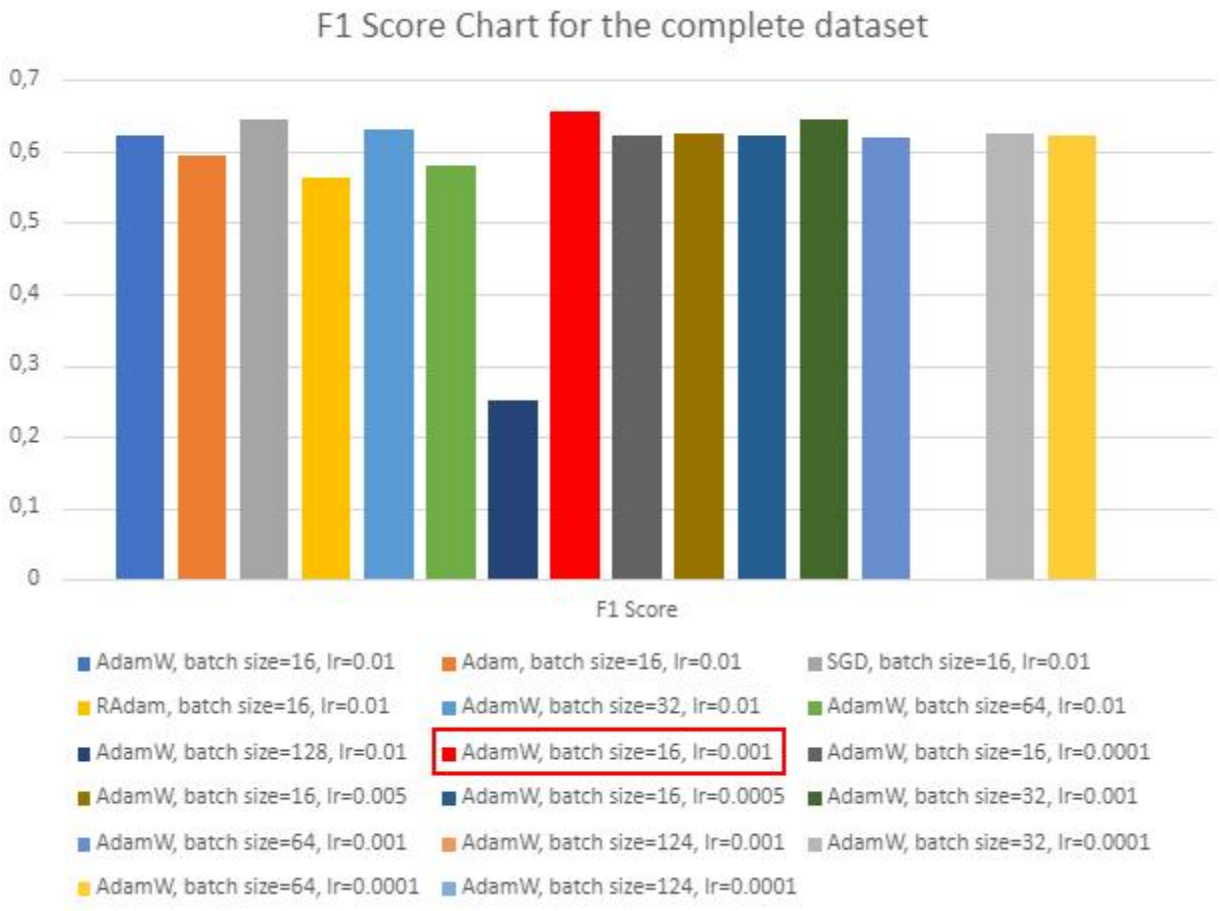

Figure 43 A cluster column diagram of the various F1 scores of our models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above

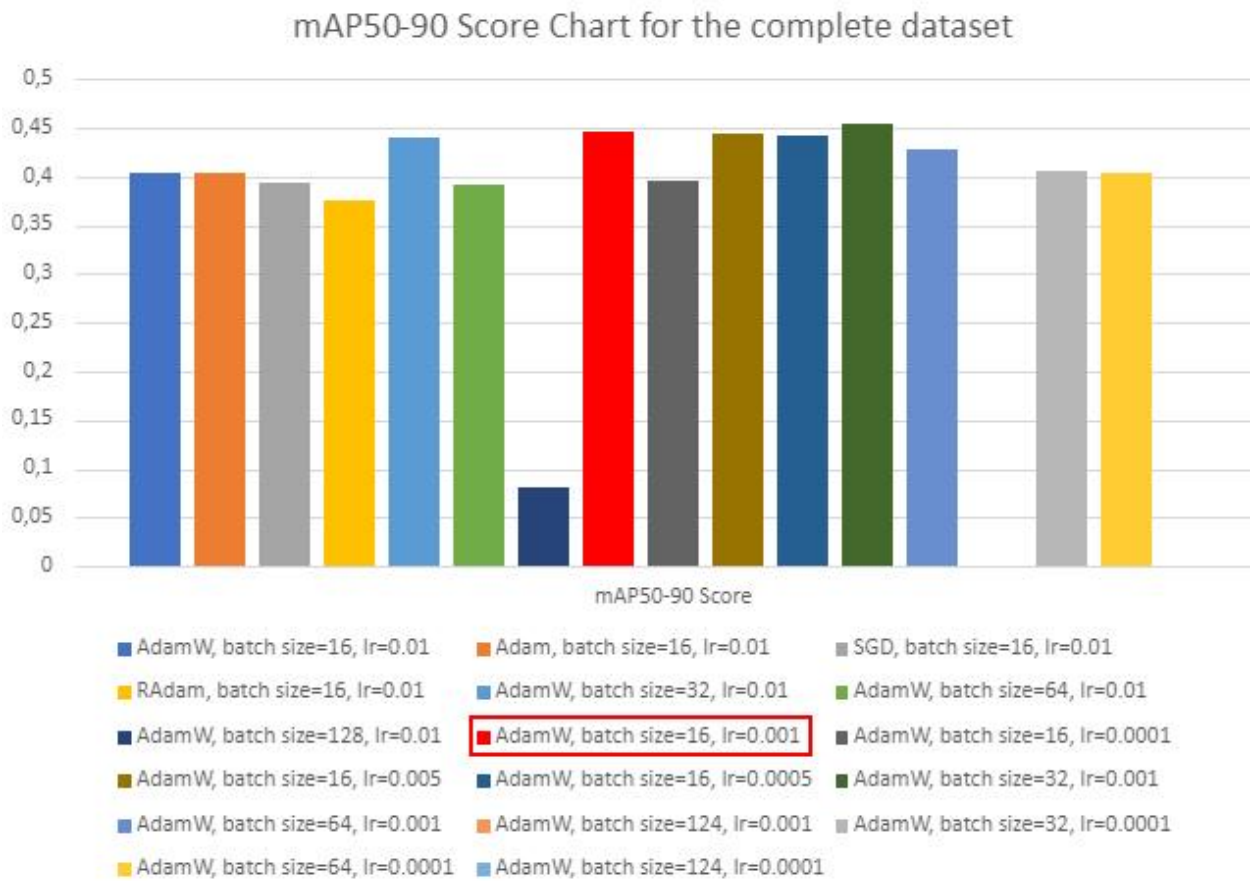

Figure 44 A cluster column diagram of the various mAP50-90 scores of our models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above

6.5 YOLONAS COMPARISON

Regarding the YOLONAS model, its training and validation process does not produce the valuable metrics that we need in order to make appropriate comparisons, unlike YOLOv8. The only metric we have is the normalized confusion matrix, which unlike the normal confusion matrix, which has absolute values, cannot be used to fabricate the precision, recall and F1 score values needed for a thorough comparison with the YOLOv8 model. That mentioned, these 2 models have been trained with the same database, meaning the truth detection rates for all the surface anomaly classes can be compared effectively.

Table 3 Comparison matrix between the most robust iteration of the YOLOv8 model we have trained and the YOLONAS model trained for 100 epochs

Metrics/Models	100 epochs, AdamW Optimizer	
	YOLOv8 batch size=32, learning rate=0.001	YOLONAS batch size=8, learning rate=0.0005
Cell	0.68	**0.79**
Diode	0.96	0.97
Cracking	**0.58**	0.4
Offline Module	**0.84**	0.26
Vegetation	**0.60**	0.52
Shadowing	0.56	0.62
Hot Spot	**0.51**	-
Hot Spot Multi	**0.57**	0.37

The observation that can be made is the fact that the YOLONAS model managed to achieve the highest detection rate for the Cell anomaly class. It outperforms our most robust YOLOv8 model by 11%, a considerable margin. We have established that the cell class is the most prominent anomaly class in our dataset, making up 31% of the images of solar panels riddled with surface anomalies. It is also the most commonly encountered solar panel surface anomaly type for solar farms worldwide [3]. Our preferred model would be one to efficiently identify an assortment of anomalies, and the YOLONAS model seems to be underperforming in the detection of other classes of anomalies, being behind the YOLOv8 model in the cracking, offline module, vegetation, hot spot, and hotspot multi classes by 18%, 58%, 8%, 51% and 20%, respectively. For perspective, the normalized confusion matrix did not appear to have a value for the hot spot category. These observations, combined with the difficulties encountered while training the model, lead us to conclude that YOLONAS is not an appropriate model for our needs.

6.6 BINARY COMPARISON

After all our attempts of fine tuning the YOLO models in order for them to be able to detect, distinguish and classify multiple solar panel surface anomalies, we prompted to train the YOLOv8 model to simply detect whether an anomaly is present. The dataset that we use for the binary identification training sessions was 90% of the whole dataset, without any exclusions. The other 10 % was used for testing. The results of these sessions are presented and compared here.

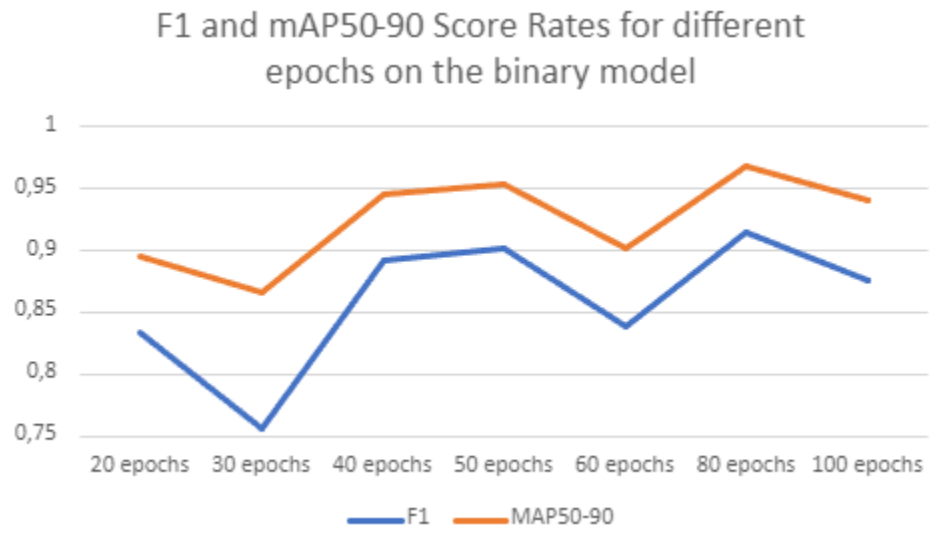

Figure 45 Figure 38 Line graph of the F1 and mAP50-90 scores Rates for different epochs on the binary dataset for the YOLOv8 nano model. The model's batch size is 16 and the model's learning rate is 0.01

When we first started to train the YOLOv8 model for 100 epochs, we were accustomed with training the multi class classification model. After the initial training cycle, we noticed an abnormality in the validation classification loss function graph, figure 40, a spike. This led us to train the model for progressively less epochs. We achieved the best performing model while training for 80 epochs, although training for half the epochs only drops the F1 score by 2.27% and the mAP50-50 by 2.17%. These small decreases

led us into comparing the truth detection rates of both models. The training cycles for less than 40 epochs had lower validation metrics, with the model trained for 30 epochs displaying a precision value of 0.64803, resulting in a lower F1 score as well.

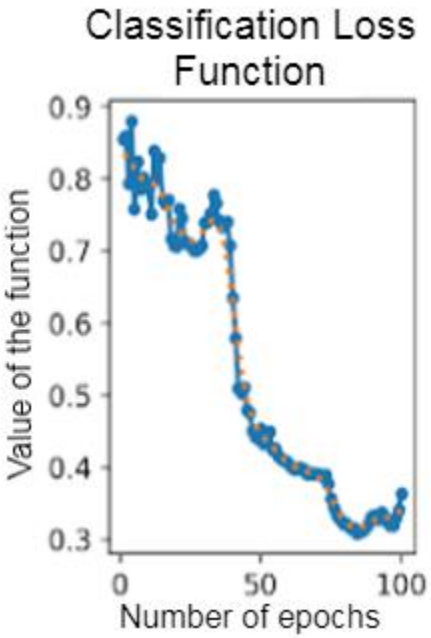

Figure 46 Validation classification loss function graph for the binary training cycle of the model YOLOv8 nano

Table 4 Comparison matrix between the binary YOLOv8 models that were trained for 80 and 40 epochs each

Metrics/ Models	YOLOv8 nano, AdamW optimizer, Batch Size = 16, Learning Rate = 0.01	
	80 Epochs	40 Epochs
Predicted + True Healthy	**0.95**	0.87
Predicted + True Faulty	0.84	0.84

This comparison indicates an 8% increase in the correct identification of healthy solar panels if the model is trained for an additional 40 epochs, which is reassuring,

considering the general rule is that more epochs equal to a more thorough training, resulting in a more robust network.

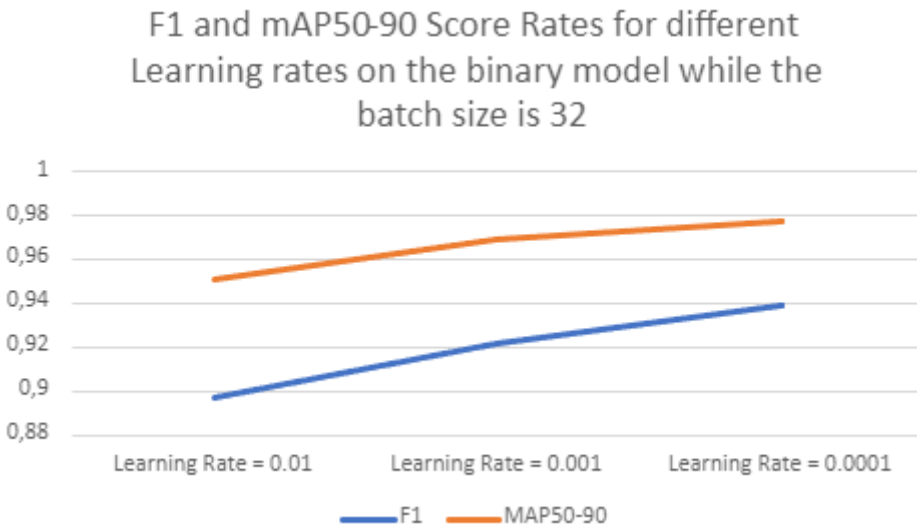

Figure 47 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the binary dataset for the YOLOv8 nano model. The model's number of epochs is 100 and the model's batch size is 16

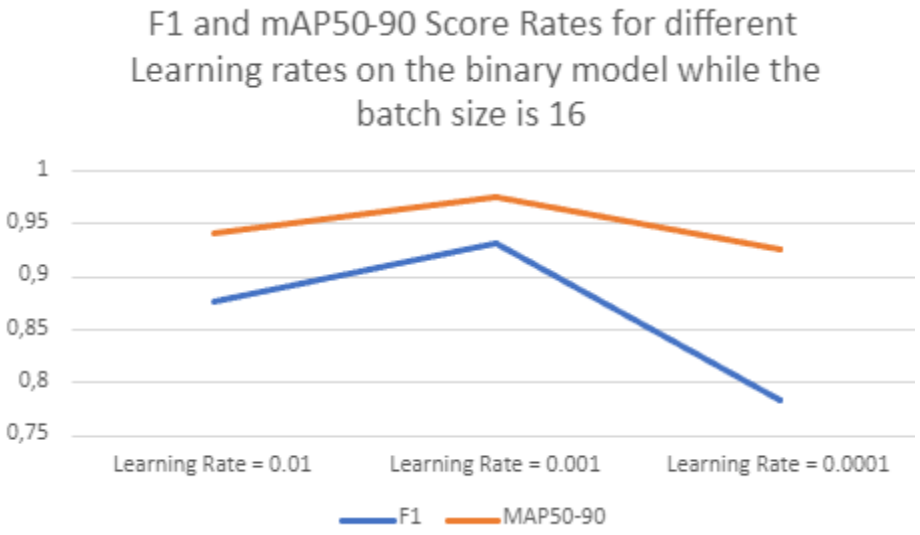

Figure 48 Line graph of the F1 and mAP50-90 scores Rates for different learning rates on the binary dataset for the YOLOv8 nano model. The model's number of epochs is 100 and the model's batch size is 32

For the purposes of this experiment, we will consider the model we trained for 100 epochs, with batch size equal to 32 and learning rate equal to 0.0001 the most competent in binary fault identification since it outperforms the previously best model by a 0.07% when it comes to F1 score and 0.02% when it comes to mAP50-90. Our most robust model outperformed our second best one in recognizing healthy panels by 1% and faulty panels by 4 percent. The differences in the validation metrics and healthy solar panel detection are indeed miniscule, while the correct identification performance gap might be noticeable between the 2 models. We proceeded by testing it, both on images originating from the same database as well as images from our augmented dataset. Its outstanding validation metrics and true detection scores both led us to this conclusion.

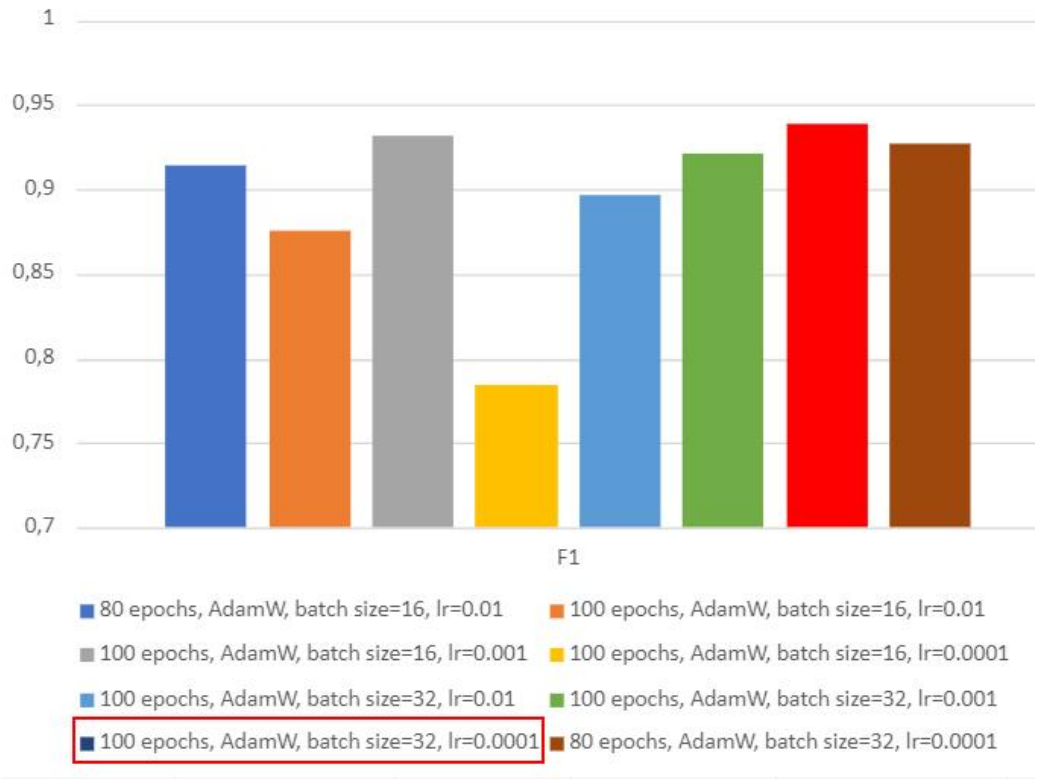

Figure 49 A cluster column diagram of the various F1 scores of our binary models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above

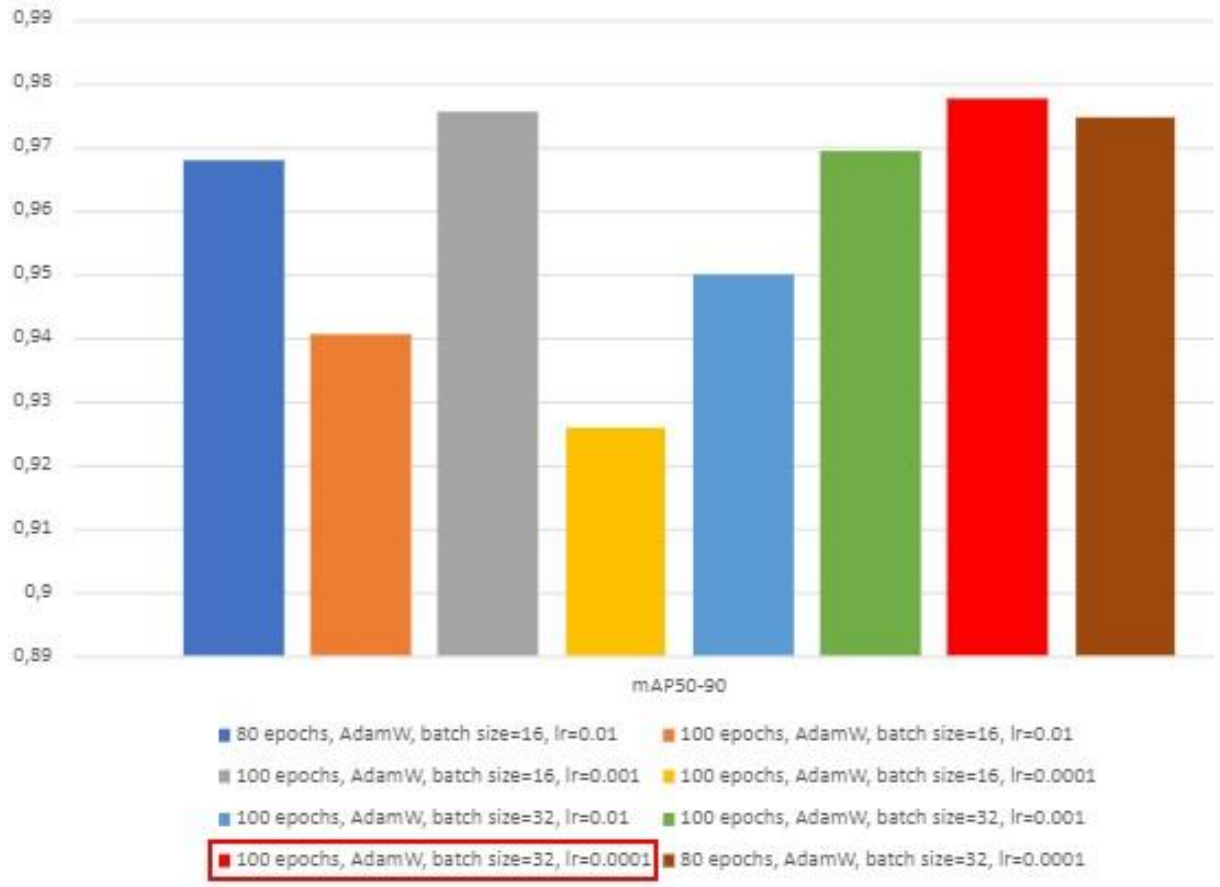

Figure 50 A cluster column diagram of the various mAP50-90 scores of our binary models trained with the complete dataset. The red column is the highest score, and it belongs to the model we discussed above

6.7 COMPARING TEST RESULTS OF OUR MOST OPTIMIZED MODELS

After the conclusions that we came to in part 7 of our thesis, we pinpointed the best optimized neural network for anomaly detection, both of multi-class classification and binary detection. These 2 selected models have been tested with the prediction function of the YOLO models. We used the remaining 10% of the database, which has not been used in either the training, or the validation of the model. This means these images were introduced to the models for the first time. The binary model has been tested also using a smaller database consisting of augmented data that we collected from the web. The average time for processing and predicting that the models need was on average 8.4 ms for the multi class classification model and 9.5 ms on average for the binary classification model. The binary classification model needed an average of 11.5 ms in order to detect the anomalies on the dataset that we created from images that we found on the web.

6.7.1 COMPARING TEST RESULTS OF OUR MULTI-CLASS CLASSIFICATION MODEL

For the purposes of this experiment, we have reserved a portion of our dataset in order to introduce our model in the testing phase. A tenth of the total images were used in order to determine the robustness of our trained models. We have mentioned that through data augmentation we managed to create a small custom dataset from the web. In the case of these images, the class of each image was never mentioned. This fact prevented us from testing the model mentioned above with the augmented dataset.

Table 5 Comparison matrix between the true detection rates of our most robust multi-class classification model in validation and testing

Metrics / Dataset	YOLOv8 small, 100 Epochs, AdamW optimizer, Batch Size = 32, Learning Rate = 0.001	
	Validation	Testing
Cell	0.67	0.77
Diode	0.97	0.87
Cracking	0.57	0.69
Offline Module	0.89	0.85
Vegetation	0.59	0.69
Shadowing	0.54	0.51
Hot Spot	0.49	0.58
Hot Spot Multi	0.53	0.70

We can assess by this comparison matrix that our model was able to quite effectively detect and classify the surface anomalies on images that it has not been trained on. Notable is the increase in cell class error detection, 10%, in cracking class error detection, 12%, in the vegetation class error detection, 12%, in the Hot Spot class error detection, 9% and in the Hot Spot Multi class error detection, 17%. The decrease in the diode class error detection, 10%, is also noteworthy. This testing process leads us to the conclusion that the model that we trained on is indeed well capable of classifying solar panel surface anomalies. Testing this model with images outside the used dataset, would further reassure us of this model's robustness.

6.7.2 COMPARING TEST RESULTS OF OUR BINARY DETECTION MODEL

Table 6 Comparison matrix between the true detection rates of our most robust binary identification model in validation and testing

Metrics / Dataset	YOLOv8 nano, 100 Epochs, AdamW optimizer, Batch Size = 32, Learning Rate = 0.0001		
	Validation	Testing	
		10% of the original Dataset	Augmented Dataset
Predicted + True Healthy	0.96	0.96	0.88
Predicted + True Faulty	0.92	0.87	0.93

It is apparent from the comparison made above that the model performs almost identically when faced with images that originated from the same dataset but have never been presented to it in the past. When it comes to images that are formatted in the shape of the dataset images, our model seems to slightly underperform in identifying healthy panels, by 8%, meaning that it falsely identifies faults when there are non-present, the least damage inducing of the 2 fault types. When it comes to detecting surface anomalies, the test on the augmented dataset reaffirms the capabilities of the network.

Chapter 7 DISCUSSION

7.1 GENERAL INTERPRETATIONS

After all the discussion in the previous chapters, the YOLOv8 model is indeed quite effective in object detection and identification. When observing the multi-class classification model, an interesting conclusion can be drawn. When we increase the amount of different anomaly classes that are introduced to the model training, the validation metrics for the model drop. Meaning that the model becomes less effective and robust, the more classes it is addressed to identify and classify. This can be attributed to the higher complexity of the task and to the imperfect quality of the images. The dataset that we used was the most readily available one publicly, but the images contained inside it were of low quality, only 24 by 40 pixels. Not much information can be effectively conveyed through such a small image, leading to loss of information, especially at the edges of the surface faults. That increases misidentification probabilities for our model. The YOLO line of models, thus YOLOv8 also, seem to excel at detecting entities when plethora of information is conveyed, when we take into consideration the fields that these models have been deployed to. Despite that drawback, our most robust model, YOLOv8 small with the AdamW optimizer, batch size equal to 32 and learning rate equal to 0.001, managed to predict efficiently faults from images that have never been presented to it before. When discussing the YOLONAS model, the acknowledgment that shall be made is that its ability for small object detection is indeed exceptional, managing to score the best Cell class anomaly true detection rate out of all the other trained models, 77%. Finally, the YOLOv8 model proved that it is an ideal model when it comes to solar panel surface fault detection from infrared imagery, as we proved by the training validation and testing of the binary detection models. This model managed to achieve both exceptional true detection rates as well as validation metrics.

7.2 COMPARISON TO PAST WORKS

In order for a better assessment of this experimentation process to be made, it is crucial that a comparison of the work in this thesis with the results of some scientific paper publications should be made.

7.2.1 COMPARISON WITH A MODEL TRAINED ON THE SAME DATASET

When we compare our experimentation process to that of this research [17], that took place in 2021 we can quickly draw some parallels between the two. The same database was used for the model training in both cases, and 2 versions of the same model were constructed, multi–class classification as well as a binary detection model. But there is where the similarities end. These researchers constructed a custom made convolutional neural network in order to approach this task. The architecture of the model is included in the scientific paper that they published. Those models did not put bounding boxes over the anomalies that they detected, it labeled the whole image and classified it as healthy or faulty for the binary or with the appropriate class for the multi–class classification model. When comparing their true detection rates to our most robust model we immediately notice that they use different classes to identify the cases of appearance of multiple cells and diodes from the cases where only one is present, a step that we excluded on purpose since our model pinpoints and counts the detected anomalies it finds in the images. Diversifying these 2 categories would only lead to more complexity and computational resource cost, without any upside. Another difference is that these researchers exclude the vegetation and cracking anomalies from their predictions, but they also have made the decision to exclude the images containing the soiling class as well, much like us. They achieve a true detection rate of 71% in the cell class, where we achieved a 77%, a rate of 92% in the diode class, where we achieved a 87%, a rate of 92% in the shadowing class, where we achieved a 51%, a rate of 58% in the diode class, where we achieved a 85% and a rate of 73% in both the hot spot and hot spot multi classes, where we achieved rates of 58% and 70% respectively. The YOLOv8 model trained for multi-class classification seems to outperform the custom CNN model in the cell and offline module classes, while being able to identify 2 more classes. It is also apparent that it is underperforming in the diode and shadowing classes of anomalies. It is important to note that these 2 models have been tested on almost the same data, with the YOLOv8 model being tested on the 10% of the total sum of the images in the database, whereas the custom CNN was tested on the 15% of the total

sum of the images in the database, as the other 85% was used on training and validation. When the comparison between the binary detection models is made, a direct look at the confusion matrixes reveal that the YOLOv8 model slightly outperformed the custom-made model, with their model correctly detecting an anomaly 93% of the time and correctly detecting the absence of anomalies 92% of the time. The YOLOv8 model has rates of 92% and 96%, respectively. The F1 score for the binary detection model was also provided in the research paper and it was equal to 0.92, being behind YOLOv8 by almost 0.18, as it had an F1 score of 0.938.

7.2.2 COMPARISON WITH A YOLOv8 TRAINED ON A DIFFERENT DATASET

We compared our experimentation process thoroughly with research that used the same database, it would be appropriate to also compare it to research using the same model. The YOLOv8 model has only been used once again in the published scientific literature for solar panel surface anomaly detection [19]. Our work differs from this research, because only a binary detection model was created, and the images used came from an electroluminescent image dataset. This study was conducted with the intention of comparing the capabilities of the YOLOv7 and YOLOv8 models. The hyperparameters that were used to train their model were an epoch count of 100, a batch size of 32 and a learning rate of 0.001. No mentions were made about the optimizer in the paper, so an assumption can be made that the default optimizer, AdamW, was used during the model's training and validation process. A direct comparison can be made this time, since this paper provides all the validation metrics for this model. It has a precision value of 0.94, a recall value of 0.91, an F1 score of 0.92, an mAP50 of 0.94 and an mAP50-90 of 0.93. The YOLOv8 model we trained achieved a precision value of 0.93, a recall value of 0.94, an F1 score of 0.93, an mAP50 of 0.98 and an mAP50-90 of 0.98. These 2 models happen to share the same hyperparameter combination. The normalized confusion matrix for the model was also provided. In this research they left the images of healthy panels unlabeled, unlike our course of action in this experiment. This model detected an anomaly 97% of the time and correctly detected the absence of anomalies 100% of the time. The YOLOv8 model has rates of 92% and 96%, respectively. The metrics suggest that our model can pinpoint and outline the present anomalies with bounding boxes efficiently, but when detection rates are under investigation, it is slightly outperformed by the model of this research.

Table 7 Studies of detecting the defects of solar cells using a deep learning approach, including our 2 tested models, which are highlighted

	Model	No. of para-meters	Number of layers	Type of image	Performance criteria	Remarks
[82]	VGG-16	0.38 M	21	EL	7.7% (balanced error), 92.3% (balanced accurate)	Considers effects of augmentation and oversampling.
[83]	DBN	4.7M	4	EL	NA	Time-consuming
[84]	CNN	1.1 M	6	EL	98.4%	Small datasets
[85]	CNN	101.2	M	27	RGB 94.3%	Considers only visible defects, not applicable for weak scratch inspection.
[86]	CNN, SVM	34.9 M	24	EL	88.42% (CNN), 82.44% (SVM)	Processing did not distinguish the defect type
[87]	CNN	0.2 M	5	EL	99.43% (SVM), 97.46% (RF), 99.71 % (CNN)	Two-category result
[88]	CNN	2.5 M	9	EL	93.02% Two-category result,	Can suffer from the over-fitting phenomenon.
[89]	CNN	12.9 M	9	EL	Micro-crack (82%), finger-interruption (81%), break (83%)	Two-category result.
[90]	CNN with Attention network & U-net	8.1 M	35	EL	99.3%	Considers hybrid loss and incorporating CNN model with other networks
[91]	CNN (LeNet)	0.06 M	7	EL	99%	Outperforms GoogleNet
[92]	R-CNN & R-FCN	-	-	EL	98.3%	The strategy of hard negative sample mining was used.
[93]	CNN	34.1 M	8	EL	F-measure 98.46%	Fuses steerable evidence filter (SEF) with the function of structural decoupling to filter the input images.
[94]	CNN (AlexNet)	61 M	25	RGB	93.3%.	Two-category result.
[95]	U-Net & Attention mechanism	-	28	EL	IOU (69%) DICE (54%)	Solves the All black issue.
[96]	CNN (ResNet50)	23.5 M	51	EL	98.59%	Used on-field low-resolution EL images.
[97]	CNN (U-Net)	-	-	EL & C-DCR	F1-score is (89%) for dark saturation current density (j0) (82%) for series resistance (Rs)	Introduces smart labeling of defects.
[98]	CNN (ResNet-50)	5.4 M	7	EL	91%	Examines the effect of different parameterizations of the normalized Lp layer on the segmentation performance
[99]	YOLOv3	-	53	RGB	94.5%	Two-category result.
	YOLOv8n	11.1 M	168	IRT	F1 Score 93.8%	Two-categoty result
	YOLOv8n	11.1 M	168	IRT	F1 Score 64.4%	8 distinct anomaly class identification and detection

7.3 LIMITATIONS

The training and validation processes for any neural network require both immense computational power and an abundance of time. Because of our limited resources, we resorted to outsourcing the processing that our model required to hardware that was provided to us by the Google corporation. This decision came with its own sets of limitations. We were allowed a specific amount of GPU time each month, limited random access memory and the worst of all, the runtime of our training was often interrupted by Google, resulting in the loss of time and available resources. On another limitation, we used only one dataset for the training of our models, meaning that we are prone to overfitting. A truly robust model ought to have the ability to process images regardless of their origin. Testing the binary model on our augmented dataset levitates that fear to some extent. Finally, these models can potentially be mounted on a UAV, a platform with limited resources available. Luckily, we took that into consideration when we chose to work with the nano and small versions of YOLOv8, which include fewer parameters in order to lighten the needed processing load.

7.4 OUTLOOK AND POTENTIAL FUTURE WORK

While looking at this project, it is natural to question the ways that can better the final results that we produced. Access to powerful hardware of our own would eliminate the dependance on cloud services, thus instantly making the quality of the experimentation process better, for the reasons mentioned above. This also ensures that the training cycle would stop due to the loss of internet signal, because the entire process would take place offline. The YOLOv8 and YOLONAS models came out in 2023, meaning that they are the forefront of image processing models in the market right now. History suggests that every model gets dethroned by a more efficient successor, the most recent example being YOLOv7 and YOLOv8 [19]. This experimentation process could be repeated once the YOLONAS model becomes more reliable, or when a new model altogether enters the market. When it comes to datasets, the one we used was the only publicly available one for solar panel surface anomalies on the infrared spectrum. If another dataset becomes available, it should definitely be used in combination with the one used in this project in order for a more robust model to be produced as a result.

Acquiring a UAV, an IRT camera and legal access to a solar farm's aerospace in order for more images to be collected might be a direction that future researchers shall explore. To conclude, loading such a model on a Jetson Series controller would be our last suggestion for furthering this work. These models are intended to be used in real time while a UAV is patrolling a solar farm, meaning that the next logical step is to try loading these models on devices that can be mounted on such vehicles. This step would enable further experimentation as many more environmental parameters would be introduced, paving the way for future researchers to further optimize these models for these specific conditions.

7.5 CONCLUSION

In this thesis we were presented with the task of proposing a novel approach in assisting the monitoring of solar farms, since the vastly expanding solar energy demand of the recent years. The use of machine learning assisted areal infrared thermography was proposed. IRT was chosen, over the more detailed EL imagery, since via aerial coverage, we can potentially set up a quick, efficient and fully automated solar panel monitoring system. EL requires manual inspection of each individual panel by a technician, a luxury when it comes to huge solar farms. The path that we took required the experimentation with machine learning models, convolutional neural networks to be exact. To the best of our knowledge the models YOLOv8 and YOLONAS have not been used before for solar panel surface anomaly multi class classification. Our goal was to efficiently train a model for binary fault identification as well as a model for multi class classification. The training of these models was possible after the manual annotation of a large AIRT dataset. After the annotation different parameters of the YOLOv8 model were changed in order for a robust model to emerge. These parameters were the model size, the number of epochs that we trained our models, the optimizer used by the models, the batch size of the models and the learning rates of the models. The YOLONAS model was also trained in this dataset. After comparing all our trained models based on a number of metrics, the most robust models emerged. We proceeded with testing these models using both some unseen to them images from the original dataset, along with images from an augmented dataset of our own creation. The result is 2 trained models that could potential be deployed on a UAV in order to assist with the automated monitoring of solar panels.

Chapter 8 REFERENCES

[1] DETECTION, CLASSIFICATION AND CHARACTERIZATION OF DEFECTS INPHOTOVOLTAIC MODULES THROUGHTHE USE OF THERMOGRAPHY, ELECTROLUMINESCENCE, I-V CURVESAND VISUAL ANALYSIS http://uvadoc.uva.es/handle/10324/42213

[2] Waqar Akram M, Li G, Jin Y, Chen X, Zhu C, Zhao X, et al. Improved outdoor thermography, and processing of infrared images for defect detection in PV modules. Sol Energy 2019;190:549–60. https://doi.org/10.1016/j.solener.2019.08.061.

[3] INFRARED SOLAR MODULE DATASET FOR ANOMALY DETECTION Matthew Millendorf, Edward Obropta, & Nikhil Vadhavkar Raptor Maps Inc https://ai4earthscience.github.io/iclr-2020-workshop/papers/ai4earth22.pdf

[4] Unmanned aerial vehicle – Wikipedia https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[5] Gallardo-Saavedra, Sara, Luis Hernández-Callejo, and Oscar Duque-Perez. "Technological review of the instrumentation used in aerial thermographic inspection of photovoltaic plants." Renewable and Sustainable Energy Reviews 93 (2018): 566-579.

[6] Al-Mashhadani, R., Alkawsi, G., Baashar, Y., Alkahtani, A. A., Nordin, F. H., Hashim, W., & Kiong, T. S. (2021). Deep learning methods for solar fault detection and classification: a review. *solar cells*, *11*, 12.

[7] Bizzarri, F.; Nitti, S.; Malgaroli, G. The use of drones in the maintenance of photovoltaic fields. E3S Web Conf. 2019, 119, 00021.

[8] Aghaei, M.; de Oliveira, A.K.V.; Rüther, R. Fault Inspection by Aerial Infrared Thermography in a PV Plant after a Meteorological Tsunami. Rev. Bras. Energ. Sol. 2019, 10, 17–25. https://doi.org/10.59627/cbens.2018.203

[9] Aghaei, M.; Grimaccia, F.; Gonano, C.A.; Leva, S. Innovative Automated Control System for PV Fields Inspection and Remote Control. IEEE Trans. Ind. Electron. 2015, 62, 7287–7296.

[10] Tsanakas, J.A.; Botsaris, P.N. An infrared thermographic approach as a hot-spot detection tool for photovoltaic modules using image histogram and line profile analysis. Int. J. Cond. Monit. 2012, 2, 22–30.

[11] Quater, P.B.; Grimaccia, F.; Leva, S.; Mussetta, M.; Aghaei, M. Light Unmanned Aerial Vehicles (UAVs) for cooperative inspection of PV plants. IEEE J. Photovolt. 2014, 4, 1107–1113.

[12] Leva, S.; Aghaei, M.; Grimaccia, F. PV power plant inspection by UAS: Correlation between altitude and detection of defects on PV modules. In Proceedings of the 2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC), Rome, Italy, 10–13 June 2015.

[13] C. M. Bishop, "Neural networks and their applications", Review of Scientific Instruments, vol. 65, pp. 1803-1832, 1994.

[14] Voulodimos, A.; Doulamis, N.; Doulamis, A.; Protopapadakis, E. Deep Learning for Computer Vision: A Brief Review. Comput. Intell. Neurosci. 2018, 2018, 7068349.

[15] S. Deitsch et al., "Automatic classification of defective photovoltaic module cells in electroluminescence images," Sol. Energy, 2019, 185, pp. 455–468.

[16] A. M. Karimi et al., "Automated pipeline for photovoltaic module electroluminescence image processing and degradation feature classification," *IEEE J. Photovoltaics*, 2019, 9, pp.1324–1335

[17] Fonseca Alves RH, de Deus Júnior GA, Marra EG, abd Lemos RP. Automatic fault classification in photovoltaic modules using Convolutional Neural Networks. Renew Energy 2021;179:502–16. https://doi.org/10.1016/j.renene.2021.07.070.

[18] Hong F, Song J, Meng H, Wang R, Fang F, Zhang G. A novel framework on intelligent detection for module defects of PV plant combining the visible and infrared images. Sol Energy 2022;236:406–16. https://doi.org/10.1016/j.solener.2022.03.018

[19] Phan, Quoc Bao; Nguyen, Tuy (2023). A Novel Approach for PV Cell Fault Detection using YOLOv8 and Particle Swarm Optimization. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.22680484.v1

[20] Wikipedia page on AI https://en.wikipedia.org/wiki/Artificial_intelligence#CITEREFGoogle2016

[21] Copeland, J., ed. (2004). *The Essential Turing: the ideas that gave birth to the computer age*. Oxford, England: Clarendon Press.

[22] Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming https://doi.org/10.1007%2F978-94-009-0279-4_9

[23] J. Hu, H. Niu, J. Carrasco, B. Lennox and F. Arvin, "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning," doi: 10.1109/TVT.2020.3034800.

[24] Application of Machine Learning Algorithms in Plant Breeding: Predicting Yield From Hyperspectral Reflectance in Soybean https://doi.org/10.3389/fpls.2020.624273

[25] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[26] D. Ciregan, U. Meier and J. Schmidhuber, "Multi-column deep neural networks for image classification," , doi: 10.1109/CVPR.2012.6248110.

[27] ImageNet Classification with Deep Convolutional Neural Networks https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[28] A. Wolfewicz, "Deep Learning vs. Machine Learning – What's The Difference?", Levity, accessed at 04/07/2023.

[29] T. Lindeberg, "Scale invariant feature transform," *Scholarpedia*, vol. 7, no. 5, p. 10491, May 2012

[30] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 886–893

[31] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[32] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *J. Physiol.*, vol. 160, no. 1, pp. 106–154, Jan. 1962.

[33] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," , doi: 10.1109/TNNLS.2021.3084827.

[34] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2016, *arXiv:1511.07122*. [Online]. Available: https://arxiv.org/abs/1511.07122

[35] D. M. Hawkins, "The problem of overfitting," *J. Chem. Inf. Comput.Sci.*, vol. 44, no. 1, pp. 1–12, Jan. 2004.

[36] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and Cooperation in Neural Nets*. Berlin, Germany: Springer,1982, pp. 267–285.

[37] Baheti, Pragati. "Activation Functions in Neural Networks." *English. In: v7labs* (2021).

[38] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 1735–1742.

[39] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 499–515.

[40] J. Yao, Y. Yu, Y. Deng, and C. Sun, "A feature learning approach for image retrieval," in *Proc. Int. Conf. Neural Inf. Process.*, 2017, pp. 405–412.

[41] H. Jin, X. Wang, S. Liao, and S. Z. Li, "Deep person re-identification with improved embedding and efficient training," in *Proc. IEEE Int. Joint Conf. Biometrics (IJCB)*, Oct. 2017, pp. 261–267.

[42] G. Wisniewksi, H. Bredin, G. Gelly, and C. Barras, "Combining speaker turn embedding and incremental structure prediction for lowlatency speaker diarization," in *Proc. 18th Annu. Conf. Int. Speech Commun. Assoc. (ISCA)*, 2017, pp. 1–6

[43] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proc. ICML*, 2016, vol. 2, no. 3, p. 7.

[44] L. Tan, K. Zhang, K. Wang, X. Zeng, X. Peng, and Y. Qiao, "Group emotion recognition with individual facial emotion CNNs and global image based CNNs," in *Proc. 19th ACM Int. Conf. Multimodal Interact.*, Nov. 2017, pp. 549–552

[45] Y. Liu, L. He, and J. Liu, "Large margin softmax loss for speaker verification," 2019, *arXiv:1904.03479*. [Online]. Available: http://arxiv.org/abs/1904.03479

[46] A Comprehensive Guide on Optimizers in Deep Learning https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

[47] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

[48] Lan, W.; Dang, J.; Wang, Y.; Wang, S. Pedestrian detection based on YOLO network model. In Proceedings of the 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 5–8 August 2018; pp. 1547–1551.

[49] Hsu, W.Y.; Lin, W.Y. Adaptive fusion of multi-scale YOLO for pedestrian detection. *IEEE Access* 2021, *9*, 110063–110073.

[50] Benjumea, A.; Teeti, I.; Cuzzolin, F.; Bradley, A. YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles. *arXiv* 2021, arXiv:2112.11798.

[51] Dazlee, N.M.A.A.; Khalil, S.A.; Abdul-Rahman, S.; Mutalib, S. Object detection for autonomous vehicles with sensor-based technology using yolo. *Int. J. Intell. Syst. Appl. Eng.* 2022, *10*, 129–134.

[52] Liang, S.; Wu, H.; Zhen, L.; Hua, Q.; Garg, S.; Kaddoum, G.; Hassan, M.M.; Yu, K. Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles. *IEEE Trans. Intell. Transp. Syst.* 2022, *23*, 25345–25360.

[53] Li, Q.; Ding, X.; Wang, X.; Chen, L.; Son, J.; Song, J.Y. Detection and identification of moving objects at busy traffic road based on YOLO v4. *J. Inst. Internet, Broadcast. Commun.* 2021, *21*, 141–148.

[54] Zheng, Y.; Zhang, H. Video Analysis in Sports by Lightweight Object Detection Network under the Background of Sports Industry Development. *Comput. Intell. Neurosci.* 2022, *2022*, 3844770.

[55] Tian, Y.; Yang, G.; Wang, Z.; Wang, H.; Li, E.; Liang, Z. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.* 2019, *157*, 417–426.

[56] Wu, D.; Lv, S.; Jiang, M.; Song, H. Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. *Comput. Electron. Agric.* 2020, *178*, 105742.

[57] Lippi, M.; Bonucci, N.; Carpio, R.F.; Contarini, M.; Speranza, S.; Gasparri, A. A Yolo-based pest detection system for precision agriculture. In Proceedings of the 2021 29th Mediterranean Conference on Control and Automation (MED), Puglia, Italy, 22–25 June 2021; pp. 342–347.

[58] Wang, Y.; Zheng, J. Real-time face detection based on YOLO. In Proceedings of the 2018 1st IEEE International Conference on knowledge innovation and Invention (ICKII), Jeju, Republic of Korea, 23–27 July 2018; pp. 221–224.

[59] Chen, W.; Huang, H.; Peng, S.; Zhou, C.; Zhang, C. YOLO-face: A real-time face detector. *Vis. Comput.* 2021, *37*, 805–813.

[60] Al-Masni, M.A.; Al-Antari, M.A.; Park, J.M.; Gi, G.; Kim, T.Y.; Rivera, P.; Valarezo, E.; Choi, M.T.; Han, S.M.; Kim, T.S. Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system. *Comput. Methods Programs Biomed.* 2018, *157*, 85–94.

[61] Nie, Y.; Sommella, P.; O'Nils, M.; Liguori, C.; Lundgren, J. Automatic detection of melanoma with yolo deep convolutional neural networks. In Proceedings of the 2019 E-Health and Bioengineering Conference (EHB), Iasi, Romania, 21–23 November 2019; pp. 1–4.

[62] Tan, L.; Huangfu, T.; Wu, L.; Chen, W. Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. *BMC Med. Inform. Decis. Mak.* 2021, *21*, 1–11.

[63] Cheng, L.; Li, J.; Duan, P.; Wang, M. A small attentional YOLO model for landslide detection from satellite remote sensing images.
Landslides 2021, *18*, 2751–2765.

[64] Pham, M.T.; Courtrai, L.; Friguet, C.; Lefèvre, S.; Baussard, A. YOLO-Fine: One-stage detector of small objects under various backgrounds in remote sensing images. *Remote Sens.* 2020, *12*, 2501.

[65] Qing, Y.; Liu, W.; Feng, L.; Gao, W. Improved Yolo network for free-angle remote sensing target detection. *Remote Sens.* 2021,*13*, 2171.

[66] Zakria, Z.; Deng, J.; Kumar, R.; Khokhar, M.S.; Cai, J.; Kumar, J. Multiscale and direction target detecting in remote sensing images via modified YOLO-v4. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* 2022, *15*, 1039–1048.

[67] Chen, R.C. Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning. *Image Vis. Comput.* 2019, *87*, 47–56.

[68] Dewi, C.; Chen, R.C.; Jiang, X.; Yu, H. Deep convolutional neural network for enhancing traffic sign recognition developed on Yolo V4. *Multimed. Tools Appl.* 2022, *81*, 37821–37845.

[69] Roy, A.M.; Bhaduri, J.; Kumar, T.; Raj, K. WilDect-YOLO: An efficient and robust computer vision-based accurate object localization model for automated endangered wildlife detection. *Ecol. Inform.* 2023, *75*, 101919.

[70] Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.

[71] Research Team. YOLO-NAS by Deci Achieves State-of-the-Art Performance on Object Detection Using Neural Architecture Search. 2023.

[72] Chu, X.; Li, L.; Zhang, B. Make RepVGG Greater Again: A Quantization-aware Approach. *arXiv* 2022, arXiv:2212.01593.

[73] Kira Ewanich, Shane Carey, and Austin Coffin. Raptor Maps 2020 global solar aerial thermography report. Technical report, Boston, USA.

[74] R.G. Vieira, F.M. de Araújo, M. Dhimish, M.I. Guerra, A comprehensive review on bypass diode application on photovoltaic modules, Energies 13 (2020) 2472. https://doi.org/10.3390/en13102472

[75] S.W. Ko, Y.C. Ju, H.M. Hwang, J.H. So, Y.-S. Jung, H.-J. Song, H.-e. Song, S.- H. Kim, G.H. Kang, Electric and thermal characteristics of photovoltaic modules under partial shading and with a damaged bypass diode, Energy 128 (2017) 232e243 https://doi.org/10.1016/j.energy.2017.04.030.

[76] Han-Chang Liu, Chung-Teng Huang, Wen-Kuei Lee, Shih-Siang Yan, Fu-Ming Lin (2015) A Defect Formation as Snail Trails in Photovoltaic Modules. *Energy and Power Engineering*,07,348-353. doi: 10.4236/epe.2015.78032

[77] F. Salem, M.A. Awadallah, Detection and assessment of partial shading in photovoltaic arrays, J. Electr. Syst. Inf. Technol. 3 (2016) 23e32. https://doi.org/10.1016/j.jesit.2015.10.003

[78] J. Teo, R.H. Tan, V. Mok, V.K. Ramachandaramurthy, C. Tan, Impact of partial shading on the pv characteristics and the maximum power of a photovoltaic string, Energies 11 (2018) 1860. https://doi.org/10.3390/en11071860

[79] C. Deline, Partially shaded operation of multi-string photovoltaic systems, in: 2010 35th IEEE Photovoltaic Specialists Conference, IEEE, 2010, 000394e000399. DOI: 10.1109/PVSC.2010.5616821

[80] V. Anand, O. Priyan, B. Pesala, Effect of shading losses on the performance of solar module system using matlab simulation, in: 2014 IEEE 2nd International Conference on Electrical Energy Systems (ICEES), IEEE, 2014, pp. 61e64. DOI: 10.1109/ICEES.2014.6924142

[81] M.R. Maghami, H. Hizam, C. Gomes, M.A. Radzi, M.I. Rezadad, S. Hajighorbani, Power loss due to soiling on solar panel: a review, Renew. Sustain. Energy Rev. 59 (2016) 1307e1316. https://doi.org/10.1016/j.rser.2016.01.044

[82] A. Bartler, L. Mauch, B. Yang, M. Reuter, and L. Stoicescu, "*Automated Detection of Solar Cell Defects with Deep Learning*," in 2018 26th European Signal Processing Conference (EUSIPCO), 2018, pp. 2035–2039.

[83] B. Ni, P. Zou, Q. Li, and Y. Chen, "*Intelligent Defect Detection Method of Photovoltaic Modules Based on Deep Learning*," in 2018 International Conference on Transportation & Logistics, Information & Communication, Smart City (TLICSC 2018), 2018.

[84] M. Sun, S. Lv, X. Zhao, R. Li, W. Zhang, and X. Zhang, "*Defect detection of photovoltaic modules based on convolutional neural network*," in International Conference on Machine Learning and Intelligent Communications, 2017, pp. 122–132.

[85] H. Chen, Y. Pang, Q. Hu, and K. Liu, "Solar cell surface defect inspection based on multispectral convolutional neural network," *J. Intell. Manuf.*, 2020, 31, pp. 453–468.

[86] S. Deitsch et al., "Automatic classification of defective photovoltaic module cells in electroluminescence images," *Sol. Energy*, 2019, 185, pp. 455–468.

[87] A. M. Karimi et al., "Automated pipeline for photovoltaic module electroluminescence image processing and degradation feature classification," *IEEE J. Photovoltaics*, 2019, 9, pp.1324–1335.

[88] M. W. Akram et al., "CNN based automatic detection of photovoltaic cell defects in electroluminescence images," *Energy*, 2019, 189, pp.116319.

[89] W. Tang, Q. Yang, K. Xiong, and W. Yan, "Deep learning based automatic defect identification of photovoltaic module using electroluminescence images," *Sol. Energy*, 2020, 201, pp. 453–460.

[90] M. R. U. Rahman and H. Chen, "Defects Inspection in Polycrystalline Solar Cells Electroluminescence Images Using Deep Learning," *IEEE Access*, 2020, 8, pp. 40547–40558.

[91] P. Banda and L. Barnard, "*A deep learning approach to photovoltaic cell defect classification*," in Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists, 2018, pp. 215–221.

[92] X. Zhang, Y. Hao, H. Shangguan, P. Zhang, and A. Wang, "Detection of surface defects on solar cells by fusing Multi-channel convolution neural networks," *Infrared Phys. Technol.*, 2020, 108, pp. 103334.

[93] H. Chen, S. Wang, and J. Xing, "*Detection of Cracks in Electroluminescence Images by Fusing Deep Learning and Structural Decoupling*," in 2019 Chinese Automation Congress (CAC), 2019, pp. 2565–2569.

[94] I. Zyout and A. Oatawneh, "*Detection of PV Solar Panel Surface Defects using Transfer Learning of the Deep Convolutional Neural Networks*," in 2020 Advances in Science and Engineering Technology International Conferences (ASET), pp. 1–4.

[95] Y. Jiang, C. Zhao, W. Ding, L. Hong, and Q. Shen, "*Attention M-net for Automatic Pixel-Level Microcrack Detection of Photovoltaic Module Cells in Electroluminescence Images*," in 2020 IEEE 9th Data Driven Control and Learning Systems Conference (DDCLS), 2020, pp. 1415–1421.

[96] A. Chindarkkar, S. Priyadarshi, N. S. Shiradkar, A. Kottantharayil, and R. Velmurugan, "*Deep Learning Based Detection of Cracks in Electroluminescence Images of Fielded PV modules*," in 2020 47th IEEE Photovoltaic Specialists Conference (PVSC), 2020, pp. 1612–1616.

[97] P. Kunze et al., "*EFFICIENT DEPLOYMENT OF DEEP NEURAL NETWORKS FOR QUALITY INSPECTION OF SOLAR CELLS USING SMART LABELING*," in Presented at the 37th European PV Solar Energy Conference and Exhibition, 2020, vol. 7, p. 11.

[98] J. Wang, B. Zhao, and X. Yao, "*PV Abnormal Shading Detection Based on Convolutional Neural Network*," in 2020 Chinese Control And Decision Conference (CCDC), 2020, pp. 1580–1583.

[99] M. Mayr, M. Hoffmann, A. Maier, and V. Christlein, "*Weakly Supervised Segmentation of Cracks on Solar Cells Using Normalized Lp Norm*," in 2019 IEEE International Conference on Image Processing (ICIP), 2019, pp. 1885–1889.

Chapter 9 APPENDIX

In this chapter the complete results of the experimentation process are presented. These results are split into categories, the same we discussed in chapters 5 and 6 of this thesis. The data is organized in 2 types of tables. The first type includes the true detection rates for the multiple anomaly categories that were extracted from the normalized confusion matrixes of the trained models. For the binary models, the complete normalized confusion matrix is presented. The data from these matrixes was used to create all the line diagrams that we presented in chapter 6. The second type of table include all the validation metrics of the trained models. From these matrixes, the F1 score and mAP50-90 score values were used in the comparisons made in chapter 6.

9.1 INTITIAL TRAINING OF THE YOLOv8 MODEL ON A SMALLER DATABASE

Table 8 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 20 epochs

Anomaly class	**True detection rate**
Cell	0.60
Diode	0.65
Cracking	0.49
Offline Module	0.76

Table 9 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 20 epochs

Metrics	
Precision	0.58456
Recall	0.60638
F1	0.59527
mAP50	0.61251
MAP50-95	0.43198

Table 10 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 50 epochs

Anomaly class	**True detection rate**
Cell	0.70
Diode	0.93
Cracking	0.62
Offline Module	0.96

Table 11 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 50 epochs

Metrics	
Precision	0.71127
Recall	0.74715
F1	0.72876
mAP50	0.76633
MAP50-95	0.55511

Table 12 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 100 epochs

Anomaly class	**True detection rate**
Cell	0.75
Diode	0.96
Cracking	0.69
Offline Module	0.95

Table 13 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 100 epochs

Metrics	
Precision	0.75586
Recall	0.81109
F1	0.78250
mAP50	0.79775
MAP50-95	0.58089

Table 14 Normalized confusion matrix of the YOLOv8 nano model trained on a smaller dataset for 120 epochs

Anomaly class	**True detection rate**
Cell	0.74
Diode	0.96
Cracking	0.64
Offline Module	0.93

Table 15 Testing metrics of the YOLOv8 nano model trained on a smaller dataset for 120 epochs

Metrics	
Precision	0.74506
Recall	0.79791
F1	0.77057
mAP50	0.79206
MAP50-95	0.58324

Table 16 Normalized confusion matrix of the YOLOv8 large model trained on a smaller dataset for 20 epochs

Anomaly class	**True detection rate**
Cell	0.44
Diode	0.45
Cracking	0.43
Offline Module	0.72

Table 17 Testing metrics of the YOLOv8 large model trained on a smaller dataset for 20 epochs

Metrics	
Precision	0.60593
Recall	0.61625
F1	0.61104
mAP50	0.64203
MAP50-95	0.46077

9.2 TRAINING OF THE YOLOv8 MODEL ON A MORE INCLUSIVE DATASET

Table 18 Normalized confusion matrix of the YOLOv8 nano model trained on an inclusive dataset for 80 epochs

Anomaly class	**True detection rate**
Cell	0.66
Diode	0.97
Cracking	0.57
Offline Module	0.92
Vegetation	0.58
Shadowing	0.43
Soiling	**0.10**

Table 19 Testing metrics of the YOLOv8 nano model trained on an inclusive dataset for 80 epochs

Metrics	
Precision	0.54272
Recall	0.62614
F1	0.58145
mAP50	0.58903
MAP50-95	0.41601

Table 20 Normalized confusion matrix of the YOLOv8 nano model trained on an inclusive dataset for 100 epochs

Anomaly class	**True detection rate**
Cell	0.65
Diode	0.97
Cracking	0.60
Offline Module	0.88
Vegetation	0.60
Shadowing	0.54
Soiling	**0.14**

Table 21 Validation metrics of the YOLOv8 nano model trained on an inclusive dataset for 100 epochs

Metrics	
Precision	0.55940
Recall	0.64213
F1	0.59791
mAP50	0.61347
MAP50-95	0.42990

9.3 DATASET MODIFICATION AND RETRAINING OF THE YOLOv8 MODEL

Table 22 Normalized confusion matrix of the YOLOv8 nano model trained on a modified dataset for 80 epochs

Anomaly class	**True detection rate**
Cell	0.68
Diode	0.97
Cracking	0.60
Offline Module	0.92
Vegetation	0.58
Shadowing	0.44

Table 23 Validation metrics of the YOLOv8 nano model trained on a modified dataset for 80 epochs

Metrics	
Precision	0.64731
Recall	0.68187
F1	0.66414
mAP50	0.68256
MAP50-95	0.47622

Table 24 Normalized confusion matrix of the YOLOv8 nano model trained on a modified dataset for 100 epochs

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.97
Cracking	0.62
Offline Module	0.92
Vegetation	0.57
Shadowing	0.52

Table 25 Validation metrics of the YOLOv8 nano model trained on a modified dataset for 100 epochs

Metrics	
Precision	0.69522
Recall	0.69651
F1	0.69586
mAP50	0.69499
MAP50-95	0.48761

9.4 TRAINING OF THE YOLOv8 MODEL ON THE COMPLETE DATASET

Table 26 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 80 epochs, AdamW optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.68
Diode	0.97
Cracking	0.53
Offline Module	0.87
Vegetation	0.58
Shadowing	0.47
Hot Spot	0.47
Hot Spot Multi	0.65

Table 27 Validation metrics of the YOLOv8 small model trained on the complete dataset for 80 epochs, AdamW optimizer, batch size=16, learning rate=0.01

Metrics	
Precision	0.65763
Recall	0.57343
F1	0.61269
mAP50	0.56169
MAP50-95	0.39898

Table 28 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.97
Cracking	0.60
Offline Module	0.92
Vegetation	0.63
Shadowing	0.52
Hot Spot	0.55
Hot Spot Multi	0.67

Table 29 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW, batch size=16, learning rate=0.01

Metrics	
Precision	0.68348
Recall	0.57218
F1	0.62288
mAP50	0.56946
MAP50-95	0.40313

Table 30 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, Adam optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.98
Cracking	0.57
Offline Module	0.86
Vegetation	0.48
Shadowing	0.40
Hot Spot	0.26
Hot Spot Multi	0.61

Table 31 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, Adam optimizer, batch size=16, learning rate=0.01

Metrics	
Precision	0.5843
Recall	0.60485
F1	0.59438
mAP50	0.57073
MAP50-95	0.40497

Table 32 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, SGD optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.69
Diode	0.98
Cracking	0.59
Offline Module	0.90
Vegetation	0.64
Shadowing	0.50
Hot Spot	0.43
Hot Spot Multi	0.75

Table 33 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, SGD optimizer, batch size=16, learning rate=0.01

Metrics	
Precision	0.61547
Recall	0.60293
F1	0.60913
mAP50	0.54946
MAP50-95	0.39313

Table 34 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, RAdam optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.66
Diode	0.96
Cracking	0.56
Offline Module	0.90
Vegetation	0.51
Shadowing	0.42
Hot Spot	0.28
Hot Spot Multi	0.55

Table 35 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, RAdam optimizer, batch size=16, learning rate=0.01

Metrics	
Precision	0.53897
Recall	0.58777
F1	0.56231
mAP50	0.54231
MAP50-95	0.37532

Table 36 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.71
Diode	0.97
Cracking	0.59
Offline Module	0.90
Vegetation	0.62
Shadowing	0.52
Hot Spot	0.38
Hot Spot Multi	0.49

Table 37 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.01

Metrics	
Precision	0.63103
Recall	0.63156
F1	0.63129
mAP50	0.63501
MAP50-95	0.44098

Table 38 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.62
Diode	0.96
Cracking	0.58
Offline Module	0.78
Vegetation	0.61
Shadowing	0.43
Hot Spot	0.30
Hot Spot Multi	0.55

Table 39 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.01

Metrics	
Precision	0.54629
Recall	0.62135
F1	0.58140
mAP50	0.57229
MAP50-95	0.39254

Table 40 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.53
Diode	0.73
Cracking	0.44
Offline Module	0.43
Vegetation	0.29
Shadowing	0.40
Hot Spot	0.06
Hot Spot Multi	0.51

Table 41 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.01

Metrics	
Precision	0.34053
Recall	0.20006
F1	0.25204
mAP50	0.12958
MAP50-95	0.08018

Table 42 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.001

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.97
Cracking	0.57
Offline Module	0.89
Vegetation	0.62
Shadowing	0.46
Hot Spot	0.57
Hot Spot Multi	0.59

Table 43 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.001

Metrics	
Precision	0.63724
Recall	0.63922
F1	0.63819
mAP50	0.62519
MAP50-95	0.44546

Table 44 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0001

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.96
Cracking	0.59
Offline Module	0.92
Vegetation	0.55
Shadowing	0.44
Hot Spot	0.32
Hot Spot Multi	0.55

Table 45 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0001

Metrics	
Precision	0.63437
Recall	0.61043
F1	0.62216
mAP50	0.62511
MAP50-95	0.39623

Table 46 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.005

Anomaly class	**True detection rate**
Cell	0.69
Diode	0.97
Cracking	0.55
Offline Module	0.90
Vegetation	0.62
Shadowing	0.48
Hot Spot	0.40
Hot Spot Multi	0.61

Table 47 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.005

Metrics	
Precision	0.59804
Recall	0.65262
F1	0.62413
mAP50	0.63048
MAP50-95	0.44413

Table 48 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0005

Anomaly class	**True detection rate**
Cell	0.69
Diode	0.97
Cracking	0.54
Offline Module	0.89
Vegetation	0.61
Shadowing	0.54
Hot Spot	0.45
Hot Spot Multi	0.59

Table 49 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.0005

Metrics	
Precision	0.58181
Recall	0.66805
F1	0.62195
mAP50	0.62532
MAP50-95	0.44306

Table 50 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.001

Anomaly class	**True detection rate**
Cell	0.68
Diode	0.96
Cracking	0.58
Offline Module	0.84
Vegetation	0.60
Shadowing	0.56
Hot Spot	0.51
Hot Spot Multi	0.57

Table 51 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.001

Metrics	
Precision	0.66433
Recall	0.62513
F1	0.64413
mAP50	0.64434
MAP50-95	0.45351

Table 52 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.001

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.97
Cracking	0.57
Offline Module	0.89
Vegetation	0.59
Shadowing	0.54
Hot Spot	0.49
Hot Spot Multi	0.53

Table 53 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.001

Metrics	
Precision	0.63048
Recall	0.60956
F1	0.61984
mAP50	0.60988
MAP50-95	0.42865

Table 54 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs with, AdamW optimizer, batch size = 128, learning rate = 0.001

Anomaly class	**True detection rate**
Cell	-
Diode	-
Cracking	-
Offline Module	-
Vegetation	-
Shadowing	-
Hot Spot	-
Hot Spot Multi	-

Table 55 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs with, AdamW optimizer, batch size = 128, learning rate = 0.001

Metrics	
Accuracy	-
Precision	-
Recall	-
F1	-
mAP50	-
MAP50-95	-

Table 56 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.0001

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.97
Cracking	0.60
Offline Module	0.89
Vegetation	0.57
Shadowing	0.49
Hot Spot	0.49
Hot Spot Multi	0.63

Table 57 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=32, learning rate=0.0001

Metrics	
Precision	0.59068
Recall	0.66273
F1	0.62463
mAP50	0.63349
MAP50-95	0.40577

Table 58 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.0001

Anomaly class	**True detection rate**
Cell	0.67
Diode	0.95
Cracking	0.55
Offline Module	0.89
Vegetation	0.60
Shadowing	0.49
Hot Spot	0.43
Hot Spot Multi	0.59

Table 59 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=64, learning rate=0.0001

Metrics	
Precision	0.61693
Recall	0.62906
F1	0.62290
mAP50	0.62946
MAP50-95	0.40468

Table 60 Normalized confusion matrix of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.0001

Anomaly class	**True detection rate**
Cell	-
Diode	-
Cracking	-
Offline Module	-
Vegetation	-
Shadowing	-
Hot Spot	-
Hot Spot Multi	-

Table 61 Validation metrics of the YOLOv8 small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=128, learning rate=0.0001

Metrics	
Accuracy	-
Precision	-
Recall	-
F1	-
mAP50	-
MAP50-95	-

9.5 TRAINING OF THE YOLONAS MODEL ON THE COMPLETE DATASET

Table 62 Normalized confusion matrix of the YOLONAS small model trained on the complete dataset for 50 epochs, AdamW optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.71
Diode	0.91
Cracking	0.47
Offline Module	0.50
Vegetation	0.57
Shadowing	0.73
Hot Spot	0.75
Hot Spot Multi	0.39

Table 63 Normalized confusion matrix of the YOLONAS small model trained on the complete dataset for 100 epochs, AdamW optimizer, batch size=16, learning rate=0.01

Anomaly class	**True detection rate**
Cell	0.79
Diode	0.97
Cracking	0.40
Offline Module	0.56
Vegetation	0.52
Shadowing	0.62
Hot Spot	-
Hot Spot Multi	0.37

9.6 TRAINING OF THE YOLOv8 MODEL FOR BINARY DETECTION

Table 64 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.91	0.21
Predicted Faulty	0.09	0.79

Table 65 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.85762
Recall	0.89464
F1	0.87571
mAP50	0.94045
MAP50-95	0.94042

Table 66 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.95	0.16
Predicted Faulty	0.05	0.84

Table 67 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.90919
Recall	0.91986
F1	0.91449
mAP50	0.96774
MAP50-95	0.96774

Table 68 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 60 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.89	0.21
Predicted Faulty	0.09	0.79

Table 69 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 60 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.81513
Recall	0.86145
F1	0.83765
mAP50	0.90238
MAP50-95	0.90193

Table 70 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 50 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.93	0.15
Predicted Faulty	0.07	0.85

Table 71 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 50 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.89791
Recall	0.90552
F1	0.90169
mAP50	0.95308
MAP50-95	0.95308

Table 72 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 40 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.87	0.16
Predicted Faulty	0.13	0.84

Table 73 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 40 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.88809
Recall	0.89548
F1	0.89176
mAP50	0.94609
MAP50-95	0.94608

Table 74 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 30 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.71	0.09
Predicted Faulty	0.29	0.91

Table 75 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 30 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.64803
Recall	0.90433
F1	0.75502
mAP50	0.86725
MAP50-95	0.86661

Table 76 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 20 epochs, batch size = 16, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.59	0.11
Predicted Faulty	0.41	0.89

Table 77 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 20 epochs, batch size = 16, learning rate = 0.01

Metrics	
Precision	0.81125
Recall	0.85709
F1	0.83354
mAP50	0.89485
MAP50-95	0.89485

Table 78 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.001

	True Healthy	True Faulty
Predicted Healthy	0.96	0.08
Predicted Faulty	0.04	0.92

Table 79 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.001

Metrics	
Precision	0.92585
Recall	0.93686
F1	0.93132
mAP50	0.97539
MAP50-95	0.97539

Table 80 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.0001

	True Healthy	True Faulty
Predicted Healthy	0.95	0.12
Predicted Faulty	0.05	0.88

Table 81 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 16, learning rate = 0.0001

Metrics	
Precision	0.67955
Recall	0.92735
F1	0.78434
mAP50	0.92581
MAP50-95	0.92585

Table 82 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.01

	True Healthy	True Faulty
Predicted Healthy	0.94	0.22
Predicted Faulty	0.06	0.78

Table 83 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.01

Metrics	
Precision	0.89043
Recall	0.90302
F1	0.89668
mAP50	0.95044
MAP50-95	0.95042

Table 84 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.001

	True Healthy	True Faulty
Predicted Healthy	0.94	0.09
Predicted Faulty	0.06	0.91

Table 85 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.001

Metrics	
Precision	0.91697
Recall	0.92627
F1	0.92159
mAP50	0.96934
MAP50-95	0.96933

Table 86 Normalized confusion matrix of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.0001

	True Healthy	True Faulty
Predicted Healthy	0.96	0.08
Predicted Faulty	0.04	0.92

Table 87 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 100 epochs, batch size = 32, learning rate = 0.0001

Metrics	
Precision	0.93724
Recall	0.94073
F1	0.93898
mAP50	0.97764
MAP50-95	0.97762

Table 88 Normalized confusion matrix of YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 32, learning rate = 0.0001

	True Healthy	True Faulty
Predicted Healthy	0.93	0.06
Predicted Faulty	0.07	0.94

Table 89 Validation metrics of the YOLOv8 nano model trained the complete binary dataset for 80 epochs, batch size = 32, learning rate = 0.0001

Metrics	
Precision	0.92179
Recall	0.93276
F1	0.92724
mAP50	0.97455
MAP50-95	0.97455

9.7 TESTING THE MOST OPTIMIZED MODELS

Table 90 Normalized confusion matrix of the YOLOv8 nano model, that was trained for multi class classification tested on the 10% of the total database

Anomaly class	**True detection rate**
Cell	0.77
Diode	0.87
Cracking	0.69
Offline Module	0.85
Vegetation	0.69
Shadowing	0.51
Hot Spot	0.58
Hot Spot Multi	0.70

Table 91 Confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the 10% of the total database

	True Healthy	True Faulty
Predicted Healthy	2886	125
Predicted Faulty	144	871

Table 92 Normalized confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the 10% of the total database

	True Healthy	True Faulty
Predicted Healthy	0.96	0.13
Predicted Faulty	0.04	0.87

Table 93 Confusion matrix of the YOLOv8 nano model, that was trained for binary classification tested on the augmented database

	True Healthy	True Faulty
Predicted Healthy	71	10
Predicted Faulty	9	154

Table 94 Normalized matrix of the YOLOv8 nano model, that was trained for binary classification tested on the augmented database

	True Healthy	True Faulty
Predicted Healthy	0.88	0.07
Predicted Faulty	0.12	0.93