An-Najah National University

Faculty of Graduate Studies

AI-BASED CRACK DETECTION AND EVALUATION IN HISTORICAL BUILDINGS

By

Leen Ibrahim Diab Abu-Shqair

Supervisors

Dr. Anas Toma

Dr. Mohammad Sammaneh

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Artificial Intelligence, Faculty of Graduate Studies, An-Najah National University, Nablus - Palestine.

2026

AI-BASED CRACK DETECTION AND EVALUATION IN HISTORICAL BUILDINGS

By

Leen Ibrahim Diab Abu-Shqair

This Thesis was Defended Successfully on 25/01/2026 and approved by

Dr. Anas Toma	
Supervisor	Signature
Dr. Mohammad Sammaneh	
Co-Supervisor	Signature
Dr. Firas Al-mahmoud	
External Examiner	Signature
Dr. Basher Tahayna	
Internal Examiner	Signature

Dedication

﴿قُلۡ إِنَّ صَلَاتِي وَنُسُكِي وَمَحۡيَايَ وَمَمَاتِي لِلَّهِ رَبِّ ٱلۡعَٰلَمِينَ ١٦٢﴾ [الانعام:162]

To my Palestinian people everywhere in the world whose strength has given me the encouragement to carry on working on this thesis during the most difficult moments, after losing my husband, Nader, who had shared this research journey with me.

To mom, dad and my dear siblings, whose love has lightened my path through the darkest times and given me the strength to stand again.

Acknowledgements

I want to thank my supervisors, Dr Anas Toma and Dr Mohammad Sammaneh for their guidance and valuable insights during this work. Their support and sometimes their ‘tough questions’ pushed me to think more deeply.

I am grateful as well to the group of students from the Civil Engineering department, including Mohammad Abuali, Jilan Tanni and Mohamad Deeb for their time and effort in annotating the dataset.

Declaration

I, the undersigned, declare that I submitted the thesis entitled:

AI-BASED CRACK DETECTION AND EVALUATION IN HISTORICAL BUILDINGS

I declare that the work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

Student's Name: **Leen Ibrahim Diab Abu-Shqair**

Signature: Leen Ibrahim Diab Abu-Shqair

Date: **25/01/2026**

List of Contents

List of Tables

List of Figures

List of Appendices

AI-BASED CRACK DETECTION AND EVALUATION IN HISTORICAL BUILDINGS

By
Leen Ibrahim Diab Abu-Shqair
Supervisors
Dr. Anas Toma
Dr. Mohammad Sammaneh

Abstract

Artificial intelligence and computer vision techniques have emerged as powerful tools for automating the process of crack detection in historical buildings. These techniques offer a robust solution for ensuring the structural integrity of historical masonry buildings. This task is essential for safety and maintenance, where proactive detection of cracks can prevent damage and risks. However, detecting cracks within masonry structures remains a challenge.

This thesis explores automated crack detection methods, where three different computer vision approaches are studied. The first approach is focused on deploying a statistical-based model that depends on mathematical fixed formulations and textural features. Then, this approach evolved into an enhanced ML-based model, where features are extracted to predict optimal detection thresholds. Finally, the third approach deployed deep learning models using transfer learning, leveraging pre-trained architectures to perform feature extraction.

To support the data-intensive requirements of the deep learning model, the research involved data collection conducted in the Old City of Nablus. Efforts were devoted to gathering data from multiple locations across the city to ensure diversity in structural conditions and crack patterns. In addition, pre-processing steps were implemented to standardize the dataset before analysis. As a result, a dataset of 2794 images of masonry structures was created. All methods were evaluated on a consistent test set to ensure unbiased comparisons.

The evaluation methodology was designed in collaboration with civil engineering experts to assess performance across two main metrics, F1-Score, and detection rate. The F1-Score reflects the reliability and detection rate reflects safety. The results showed a clear

hierarchy in performance. The statistical-based crack detection model achieved a baseline F1-Score of 0.4775. The ML-based model improved with an F1-Score of 0.55151, confirming the advantage of data-driven parameter optimization by ML. The single class training of the YOLOv8 model showed good reliability, achieving the highest F1-Score of 0.6116, effectively balancing precision with sensitivity. However, when evaluated on the detection rate, the single class training of the Mask R-CNN proved superior performance identifying 89,23% of all potential cracks.

Keywords: Crack Detection, Historical Structures, Image Processing, Machine Learning, Deep Learning, Transfer Learning.

Chapter One
Introduction

The integration of artificial intelligence into cultural heritage preservation represents an emerging field within structural maintenance. Advanced computer vision techniques are applied to protect historical sites and to mitigate potential risks. Historical buildings are structures that have outstanding architectural and historical value due to their age, style or association with important events in history. Historical buildings can include a variety of structures, such as palaces, mosques, churches and theaters. Masonry construction, in particular, has been used in heritage structures. It represents one of the dominant techniques used in historical buildings. It is a type of building system in which the external supporting walls are completely made of masonry units beside filling mortar. The masonry walls carry the building loads and provide primary structural support for all major components.

One historical area that has a lot of history is the Old City of Nablus, located in Palestine. The city is filled with a large number of traditional masonry structures including old mosques, churches, khans, and residential buildings. Figure A.1 shows a map that displays the distribution of various traditional structures across the old city.

Historical buildings are often challenged by climate and natural factors, including exposure to environmental factors, vibrations, earthquakes and more [1]. As a result, many of these structures experience varying degrees of deterioration from minor to severe. All of this makes it really crucial to keep historical buildings safe by assessing their conditions constantly [2], not just to make these buildings safe but also to preserve their integrity and value. Societies can protect their identity, because heritage is more than just buildings. It reflects dreams, cultures and memories [3]. The reuse of historical buildings can attract tourists, while preservation drives economic development through heritage tourism [4].

The Old City of Nablus is a good case study of urban heritage preservation [3]. The Old City has faced many natural threats including two major earthquakes. The first one in 1834, which led to the collapse of several buildings from the Ayyubid and from the Mamluk periods. The second in 1927, which caused damage to the city’s structure [5]. Despite these events, Nablus has managed to preserve much of its historical fabric

through continuous restoration and conservation efforts [6]. At the same time, not all areas have been preserved the same way, some structures still show signs of damage and deterioration. Such as, cracks. This has created a valuable opportunity for this study to collect data for the task of crack detection. Moreover, the overlapping of the ancient Canaanite, Roman, and Ottoman structures has shaped a unique style [7], [8]. All of these reasons built a good foundation for creating the dataset used in this research.

It is important to understand that the multi-layered construction techniques used in historical buildings are really complex. Diverse materials and cumulative repairs also make these buildings complex. These complexities make it challenging when it comes to evaluating historical buildings, require special methods and cause many difficulties for the researchers [9]. Cracks are important indicators of potential structural damage in historical buildings, information on cracks is essential for the assessment of structural safety [10]. An analysis of 50 reports executed between 2000 and 2015 by the Civil Engineering Faculty of Porto University on heritage constructions built in the time period between the 16th and 20th centuries showed that "walls and arches are the most affected elements when damage analysis was done, and that 85% of these cases were observed with cracks" [1]. This exactly explains why detecting and evaluating cracks is needed to support the preservation of historical buildings. Crack detection enables a full understanding of the potential risks these structures may face, this allows for proactive measures to be taken.

1.1 Theoretical Basis

Several agencies like the Federal Emergency Management (FEMA) have developed many codes and specifications for rapid evaluation of historical buildings, such as the visual inspection [11]. Civil engineers have relied on manual inspection for crack detection. However, this method is costly and subject to human error due to its dependence on subjective judgment [12]. It is also inefficient in urgent scenarios where fast assessments are critical.

As cameras became more accessible, computer vision methods emerged as a more practical and cost-effective solution [13]. Features for crack detection can be extracted by image-based approaches using two main methods: image processing (IP) & machine learning (ML) architectures. Figure A.2 shows an overview of the crack detection approaches discussed in the literature, which also outlines the process from visual

inspection to image processing and further through ML methods, including both deep learning techniques and advanced deep learning approaches like transfer learning.

Image processing and machine learning techniques can enhance the efficiency and the speed of manual assessments especially during catastrophes when it is needed to evaluate large datasets quickly. These assessments have relied on image processing techniques that include selecting and extracting images features manually like edges, textures and patterns [13]. These techniques may produce inaccurate results when there is a variation in the typology of the building or variations in the materials, designs, or environmental conditions [14]. To move beyond the limitations of manual feature selection, these image processing techniques can be integrated with machine learning models that are capable of learning and extracting relevant features from the data automatically. This integration allows the system to identify complex patterns that can't be easily detected by the human eye.

Although machine learning methods have demonstrated improved accuracy against traditional statistical methods, they still face a significant limitation. A strong dependence on manually designed features, which can limit their ability to capture complex relationships within cracks [15]. On the other hand, deep learning techniques overcome these limitations by learning to extract features directly from the data. These models can be very resilient to noise. This allows them to detect cracks even in challenging conditions like inconsistent lightning or in different structural types [16].

Deep learning has shown significant success in crack detection, learning to many studies on deep learning crack detection approaches. A recent paper [17] provides analysis of deep learning-based crack detection methods, showcasing the contributions of these methods have added to this task. The primary challenge for deep learning models is their dependence on large datasets to achieve an effective performance [18]. To address this challenge, transfer learning (TL) has come to light as a powerful approach. Transfer learning is the process of developing a model for a specific problem and then applying it in some form to address related secondary problems as shown in Figure A.3. It is useful and advantageous when dealing with limited data, because it allows to leverage the knowledge gained from the primary problem to improve performance on tasks with smaller datasets. Transfer learning also reduces the time required to train the model.

Utilizing the advantages provided by transfer learning, this study [18] provides a comparison of various transfer learning frameworks that can be applied to crack detection and classifications. By using these frameworks, even with limited data, the models can classify cracks in concrete walls, achieving high accuracy.

1.2 Problem Statement & Study Objectives

Although crack detection has been extensively studied, limitations and challenges remain. Much of the existing work focuses only on the binary crack classifications. These studies aim to tell if a crack is present in an image or not. However, such approaches remain limited and not focused on crack-level analysis. On the other hand, the nature of the crack detection task itself is really complex due to crack characteristics especially in historical buildings. Most studies have focused on straightforward images and datasets, while this research deals with complex masonry structures. This category of historical masonry buildings has received little attention in the literature, identifying a clear research gap.

The dataset that is created presents unique difficulties, it involves structure with varying conditions, and noise that can be considered as deterioration. Cracks themselves are also very challenging to be detected as they are close to brick components. Figure 1 shows data samples from our dataset, highlighting these complexities. This dataset covers a domain that has not been explored yet.

Figure 1

Samples from the dataset

The primary goal of this study is to conduct comparative research, from developed statistical-based models to using transfer learning models. This comparison will allow us to identify the most effective approach for this specific problem. To achieve this, the study will focus on the following objectives:

1. To develop crack detection models in historical buildings, starting with a statistical-based model through machine learning based model to transfer learning.
2. To explore the potential of transfer learning in improving the accuracy of crack detection.

3. To evaluate and compare the accuracy of each method based on its ability to identify cracks in complex historical masonry structures.

1.3 Study Hypothesis

This study is driven by the following main hypotheses:

1. Computer vision algorithms can provide a fast, efficient and cost effective approach for analyzing large-scale datasets, enabling the automated detection and assessments of cracks in historical buildings.
2. Using the machine learning based model for crack detection will achieve a higher accuracy than the statistical model, i.e.,Using more features will improve the results.
3. Deep learning models will exceed both the statistical and the machine learning based methods used in this research.

The validation of these two hypotheses is expected to prove that the efficiency of crack detection in historical buildings will be enhanced through data-driven techniques. It is also expected that the use of transfer learning will overcome the challenges raised by complex and limited data, leading to a reliable identification of cracks.

1.4 Thesis Contribution & Importance of the Study

The value and the importance of this study lies in its structured comparison of three automated crack detection approaches in historical buildings. Including developing a statistical-based model using image processing, then a machine learning based model, and finally by leveraging transfer learning on deep learning models. By evaluating these models on the same test set using F1-Score, this research provides an answer to which model is most effective for this challenging task. Another core contribution of this thesis is the creation of the labeled dataset. It is called NADER-Crack and it provides a valuable benchmark for future research including 2794 images, annotated by collaboration with the Department of Civil Engineering at An-Najah National University. This collaboration provided assistance in passing the knowledge and the deep needed understating of historical structures. Domain experts in this department ensured that the developed system is contextually relevant and accurate.

1.5 Thesis Outline

This thesis is structured as a journey through the development of automated crack detection methods for historical buildings.

- Chapter 1: The problem, motivation and hypotheses are introduced.
- Chapter 2: Exploring previous research, highlighting the gaps that lead to this research.
- Chapter 3: Explains the core ideas behind the concepts used in this research, to help the reader follow the technical path ahead.
- Chapter 4: Walks through the methodologies step by step.
- Chapter 5: Describes how the experiments were set up and how models were trained.
- Chapter 6: Results are presented and discussed.
- Chapter 7: Reflects on the key findings, limitations and suggests future work.

Chapter Two
Literature Review

Crack detection has been applied across various fields and contexts like streets, pavements, bridges and other surfaces. Each use case requires unique techniques for detecting and evaluating cracks. It is crucial to study the methods used in these different contexts to develop the best model for crack detection specifically in masonry addressing the specific challenges of historical structures.

Many approaches have been applied to the crack detection task, starting with image processing, morphological techniques have been implemented for this purpose due to its effectiveness in identifying features with specific shapes. An example of this, is the study conducted by [19]. This study focused on using morphology operations for steel slabs crack detection, the method segmented the data using mathematical morphology and employed a logistic regression model to classify the identified regions as cracks or not cracks.

A similar image processing technique was used but for detecting concrete surface cracks in the research by [20]. Initially, a morphological technique was used to correct and adjust the brightness of the background, enhancing thresholding that was used in next steps, to improve detection performance. Then, a detailed algorithm was applied to calculate crack length and width. In addition to that, a neural network was developed to classify crack patterns, to tell if the crack is horizontal, vertical, diagonal or random. The results showed that this technique provided a high accuracy for analyzing the crack characteristics.

The foundation of the methodology presented in these two papers refers back to the revolutionary work in image analysis using mathematical morphology introduced by Haralick, Sternberg, and Zhuang in their 1985 paper [21]. This paper introduced the use of mathematical morphology in image analysis for the purposes of object or defect identification, which has significantly influenced the development of crack detection methods in the subsequent research.

Expanding on the use of image processing techniques in crack detection, recent research [22] introduces an innovative image processing framework for the numerical modeling of cracked masonry structures, focusing on micro-scale models. In detailed micro-modeling (Figure A.4 - b), masonry is represented as a collection of individual units and

mortar joints, with bricks and mortar modeled using continuous elements, while the interactions at their interfaces are depicted using discontinuous elements [23], [24].

A watershed-based algorithm was proposed in [22]. It begins by breaking down the segmented features of the masonry structure. These features are then reassembled into shared vertices and edges, which are next converted into polylines that are scalable. A contour generalization algorithm will be applied to these polylines to simplify them. Finally, geometric adjustments are made to the masonry elements to ensure transition to the numeral model. After applying this algorithm, the mortar, mortar damage, and block damage were identified. Structural features were presented, allowing for detailed analysis of the damage. The final results were compared to an idealized reference model to assess its effectiveness and found to be effective for many applications like crack detection. However, a key limitation is that it only assesses structures with similar behavior, rather than predicting the behavior of structures. It also depends on a regular grid pattern for comparison, which presents a challenge when working with a dataset like ours, where collected images lack a consistent pattern.

It was proven that it is more efficient to integrate machine learning with image processing models for crack detection. A similar approach was adopted by Hyeong-Gyeong and Jung-Hoon (2011) [25], where they combined image processing with machine learning for image classification. The images were utilized as inputs to train a neural network after being processed by image processing techniques. Data was structured into a matrix with each row showing extracted features in the image. The neural network included 5 hidden layers, each composed of 5 neurons. The output layer had a single neuron that worked as a binary classifier, to classify the image as cracked or not cracked. Yet, this study concluded that more improvements are needed to achieve better results.

In recent years, deep learning methods have gained popularity as a replacement for machine learning approaches for crack detection. CNNs are the foundation of these methods [26], with many algorithms and models being developed on top of its architecture to improve accuracy and speed. Deep learning has been used in many studies for concrete crack detection [27], [28]. Comparing CNN's results with image processing techniques was studied in [29], instead of relying on image processing only like in [22], it was used as a final step after using machine learning for detecting cracks on brick masonry walls with a regular pattern, leading to better results. The study also compared

various CNN architectures like U-Net, DeepLabV3+, LinkNet and FPN, originally implemented in this paper [30].

Chaiyasarn et al. (2018) [31] proposed a crack detection system for detecting cracks in masonry structures in historical sites. The system combined deep convolutional neural networks (CNNs) with support vector machines (SVMs). This model performance was better than using CNNs only. However, this performance depended on the availability of a large dataset, which can be a limitation in practical applications.

Faster R-CNN (faster region proposal convolutional neural network) is another well-known architecture used in the crack detection task. This architecture consists of two main components. The first one is a feature extraction network (FEN) and the second is a regional proposal network (RPN). These two networks operate using a shared component. This enables real-time image processing [32]. The motivation behind the shift to Faster R-CNN is the limitations of standard CNNs, especially their tendency to produce partial detections. This study [33] used Faster R-CNN for crack detection, it was trained for detecting cracks on concrete surfaces. An innovative "Progressive detection" method was developed and showed high accuracy. This method depends on splitting the image into chunks then running the detection by Faster R-CNN on each chink, after that, it merges all the chunks again.

Transfer learning was also used in crack detection, addressing the challenge of limited data availability. Dais et al. (2021) [30] was one of the earliest studies exploring deep learning for pixel-level crack segmentation on masonry structures. It applied transfer learning for both crack classification and segmentation, with pixel-level segmentation being a key method, rather than block-level classification for crack detection. In this paper, the encoder-decoder architecture was used and compared with both transfer learning and deep learning methods across various networks. Transfer learning methods achieved superior results. U-Net - a deep fully convolutional network (FCN) - with MobileNet and FPN, a general pyramid representation network [34] with InceptionV3 achieved the highest F1-Scores, outperforming other networks for crack segmentation found in the literature.

Researchers have also explored hybrid approaches for crack detection, where multiple algorithms are combined. These hybrid models have gained popularity recently, due to their ability to take advantage of the strengths of each algorithm, improving the overall

performance. A hybrid concrete crack segmentation model was implemented in [35], the model addressed images with complex backgrounds by first implementing the Faster R-CNN to detect crack regions. Then, the modified tubularity flow field (TuFF) algorithm was used to segment crack pixels in these detected regions. To measure the crack thickness and length in pixels, a modified transform method was implemented. Integrating all of these models together contributed to the overall enhancement in detecting cracks. Another paper, titled "Crack45K" [36] used TuFF with a vision transformer and also used a sliding-window algorithm for segmenting cracks in pavement structures. The study title was inspired by the dataset used, the dataset contained 45K images. The study achieved good results implementing all of these algorithms together.

An important study to mention, is the one that compared YOLOv8 and Mask R-CNN deep learning models for instance segmentation across two datasets for complex environments. This paper [37] concluded that YOLOv8 achieved higher accuracy when it came to real-time applications compared to Mask-R-CNN. Another paper [38] also compared the same exact models but specifically for concrete crack detection, and concluded that Mask R-CNN was better for this specific task.

Chapter Three
Foundational Concepts

This chapter will explain the core theory behind the computer vision methods used in the thesis. Each section presents the essential theoretical concepts required to understand the design and functionality of the pipelines detailed in the methodology.

3.1 Digital Image Processing Principles

3.1.1 Gray-scale Conversion

The image is represented as a two-dimensional array or a matrix, where each element in this array or matrix represents a pixel. For color images, three components are required for each pixel to represent the color: red, green and blue. In some cases, the image is transformed to gray-scale, where the three color components are converted to one component called gray level. This technique is used when the region of interest is identified by its shape not by its color, the color may not have significant information. The weighted sum method is used to translate the input images from color space to gray scale according to the following equation:

$$\text{Gray} = 0.299\text{R} + 0.587\text{G} + 0.114\text{B} \qquad \text{(Eqn. 1)}$$

Where R is the red, G is the greed and B is the blue component of the pixels. Weights reflect sensitivity of the human eye to the three-color components. This method maintains the visual information of image contents by preserving the relative variations between all of the gray levels [39].

3.1.2 Gaussian Blurring

Gaussian blurring filter contributes to reducing the noise in the image [40]. The process is performed by applying the moving kernel shown in Table B.1 in Appendix B on all pixels of the image. The new value is calculated by the summation of the multiplication between the kernel coefficients and the corresponding pixels.

3.1.3 Segmentation & Thresholding

Image segmentation is one of the computer vision techniques that partitions an image into groups of pixels that have similar characteristics to inform detection tasks. Many methods

can be used in image segmentation, but the most popular one is thresholding. Thresholding techniques divide the image pixels according to their intensity level, it creates new images classifying pixels based on whether their intensity is above or below a given "threshold value" Thresholding is used in various applications like character recognition, object detection, and image segmentation, enabling efficient analysis of digital images [20], [41].

If the threshold value is selected manually, then this is called static thresholding. But, if one of the popular methods like otsu's method is used, then it involves finding an optimal threshold value based on pixel intensities in the image. Local thresholding, however, calculates different threshold values based on local pixel intensities for each region instead for the whole image at once. A good example of this is the Adaptive gaussian thresholding [42].

In static thresholding, the gray-scale image is segmented based on a thresholding value T, where the objects with intensity levels greater than T are assigned to 1 and the remaining levels are assigned to 0, the output image $g(x, y)$ can be obtained by the original input image $f(x, y)$ as shown in the equation below:

$$\mathrm{g(x,y)} = \{\, 1, \mathrm{if\ f(x,y)} \geq \mathrm{T} \qquad 0, \mathrm{if\ f(x,y)} < \mathrm{T} \,\} \qquad \text{(Eqn. 2)}$$

Otsu's method can be used to perform image thresholding automatically. This method outputs a single intensity threshold like static thresholding but this value is found automatically by minimizing intra-class intensity variance, or by maximizing inter-class variance [43], [44].

Adaptive gaussian on the other hand is an effective thresholding used for images with inconsistent light [42], [45], [46]. The method uses an odd integer as the block size, allowing each region to be assigned a distinct threshold value based on its local variations. A constant 'C' is then subtracted from the mean to produce an appropriate thresholding result [42], [45], [46]. Thresholding is found by the following formula:

$$\mathrm{T(x,y)} = \mathrm{m(x,y)} - \mathrm{k} * \mathrm{s(x,y)} \qquad \text{(Eqn. 3)}$$

where T is the threshold value at pixels x and y, m is the local mean of pixels in the block, s is the local standard deviation, and k is the constant defined by the user that controls the sensitivity of the thresholding process.

3.1.4 Morphological Operation

Morphological operations are a group of image processing techniques that help refine the shapes and structures found in an image. They are grounded in mathematical morphology, a field that studies how shapes and patterns behave. In practice, these operations are used to clean noise, adjust, or highlight specific features in an image, and play an important role in image processing tasks. The most common operations are erosion, dilation, opening and closing [47].

In this study, closing is used in the image processing pipeline in both statistical and ML based models, to close small holes and gaps within the foreground objects while preserving their overall shape. Closing combines dilation followed by erosion. While erosion shrinks objects in a binary image, dilation expands them. A structuring element is used to perform erosion and dilation operations, as defined by the following equations:

$$\text{Dilation: } g(x,y) = \{\, 1, \text{if s Hits f} \qquad 0, \text{otherwise} \,\} \qquad \text{(Eqn. 4)}$$

$$\text{Erosion: } g(x,y) = \{\, 1, \text{if s Fits f} \qquad 0, \text{otherwise} \,\} \qquad \text{(Eqn. 5)}$$

where f and g are the input and output images respectively, hitting refers to the condition where all the "1" pixels in the structuring element overlap with at least one foreground pixel in the binary image. In contrast, fitting occurs when at least one pixel in (s) aligns with a foreground pixel in the binary image [39].

3.2 Machine Learning for Regression

Regression analysis is a supervised machine learning technique that is used when the goal is to predict continuous numerical values by modeling the relationship between independent variables (features) and a dependent variable (outcome) [48]. The trained model in our methodology will take the feature vector of unseen test images and outputs an exact numerical prediction for its optimal threshold. To identify the most effective algorithm for the task of finding this optimal output, many regression algorithms are evaluated in this study [49]. Through this investigation, the three models detailed below are selected and tuned to train with the best performing one. The reason behind selecting these three models is to provide a comprehensive and a robust evaluation.

3.2.1 Random Forest Regressor

Random forest regression is a powerful baseline supervised machine learning algorithm. It is based on the principle of ensemble learning. By using the bagging technique, it combines and averages the predictions of multiple decision trees to produce a more accurate final prediction. This way, it prevents overfitting by having some randomness [50]. Figure A.5 illustrates the architecture of this algorithm.

3.2.2 Extreme Gradient Boosting (XGBoost)

Extreme gradient boosting is another ensemble method, but instead of building many models in parallel, boosting builds models sequentially, where each new model is trained to correct the errors made by the previous ones as shown in Figure A.6. By this technique, XGBoost can build highly accurate predictive models and it is famous for its efficiency [30], [31]. This makes it a good choice for tasks involving structured or tabular data, such as the feature set employed in this study.

3.2.3 Support Vector Regression (SVR)

Support vector regression is a machine learning algorithm that is built on top of the support vector machines (SVMs) concept, but extends it to regression tasks. This concept is based on finding a function that best fits the data while tolerating a certain degree of error [51]. A margin of tolerance called the epsilon (ε)-tube is defined around the function as shown in Figure A.7.

SVR can handle non-linear relationships by employing the kernel trick. It is a function that maps features into high-dimensional spaces. By this mapping, and its margin based approach, SVR can capture hidden patterns in the data [52]. It is important to include another model that is not a tree-based model, alongside the two models above, to ensure a holistic comparison.

3.3 Deep Learning for Instance Segmentation

Deep learning has significantly advanced the capabilities of digital image processing and machine learning, extending far beyond what was previously achievable. Instead of depending on manually selected features, deep learning models can automatically learn from raw pixel data. Deep neural networks made this possible in all tasks and especially in computer vision tasks, introducing the end-to-end learning, where the model is just

given a big dataset of images with annotations for each image. Deep learning needs big data [53]. Millions of data records are required to achieve good results depending on the task. A good example of a common dataset that is referred to in this study is: Common Objects in Context (COCO) [54]. It consists of 2.5 million images with 91 object categories. For the crack detection task in this study, assembling such a massive dataset is often impractical. To address this challenge, Transfer learning (TL) techniques will be applied.

The idea behind transfer learning is to take a model that has been pre-trained on a general purpose dataset, that is a very large dataset (such as COCO) and adapt it for a new task where it is going to be specific and will have a smaller dataset [55]. That way, instead of learning from scratch, the model leverages existing knowledge where it has already recognized many of the visual features from the initial training. Transfer learning requires fine-tuning to achieve its goal for the specific task, which will be discussed in detail in the methodology section.

3.3.1 Convolutional Neural Networks (CNNs)

One of the most used deep learning algorithms in computer vision is the convolutional neural network (CNN). They are primarily employed in image classification, but also play a central role in segmentation and object detection [56], [57]. It processes images by a series of layers to learn more complex features from each layer. The deeper the layer, the more complex features are being learned [58]. The core layer is the convolutional layer. In this layer, a small filter (or kernel) slides over the input image to create a feature map, highlighting specific patterns.

3.3.2 From Classification to Instance Segmentation

Many computer vision tasks can be used for the specific task of crack detection. It is important to do research and differentiate between these tasks, to justify the selection of the final models. Each one of these tasks offers a different level of detail.

- **Image Classification**

In image classification, the task involves looking at an entire image, and assigning a single label to it, without looking into its details. An example of this for our case, the output

would be, an image classified as “Contains Crack” or “No Crack”. This method is not selected because it is too simple for the research goal.

– **Object Detection**

The task of object detection means that the model identifies the location of objects and draws a rectangular bounding box around each one, with a class label. For our example, a bounding box is drawn around a crack with the label “Crack.” This provides a location but with a very poor presentation of the exact position, or the true shape of the crack.

– **Semantic Segmentation**

A class label is assigned to each and every pixel in the image in a semantic segmentation task. If two cracks are close to each other, semantic segmentation would merge them into a single region, treating them as one crack. This will prevent counting or analyzing separate cracks in this study case.

– **Instance Segmentation**

Instance segmentation is the most powerful and detailed task from all tasks for our goal. It combines the benefits of object detection and semantic segmentation, where the model can identify every separate object instance and provides a precise, pixel-level mask for each one instead of a bounding box. The bounding box only gives a total area including parts of the background. But in our case, if there are two separate cracks, the model will produce two separate, pixel-level masks, each labeled “Crack.” This level of detail is the ideal choice for our research. It confirms the presence of a crack in an image and shows its exact locations, and also provides the shape and pixel count for each individual crack. For instance segmentation, Mask R-CNN and YOLO are the most commonly used models.

– **Mask R-CNN**

Mask Region-based Convolutional Neural Network (Mask R-CNN) was first introduced in 2017 [59]. It extends the success of Faster R-CNN to instance segmentation, and both of them are considered to be two-stage architectures [32]. To understand this architecture, it is important to understand its base. R-CNN is a machine learning family in computer vision used for object detection. Its final result creates a bounding box around the regions of interest. Faster R-CNN is the next version of it, it does a better job by incorporating

attention mechanisms introducing a Region Proposal Network (RPN). Attention mechanism is the concept of allowing neural networks to prioritize specific parts of the input data while filtering out less relevant information. RPN is a small network that achieves this concept by sliding over the feature map generated by the backbone network first in Faster R-CNN, to propose regions that have a high chance of containing objects. In the second stage of Faster R-CNN architecture, adaptive pooling extracts features from each region defined by proposal boxes that more precisely represent the region's features.

This is done by RoI pooling. RoI pooling extracts fixed-size feature maps from variable-sized regions of interest (RoIs) within an image's feature map. Max pooling is performed to achieve this goal. Faster R-CNN delivers strong performance in object detection, but there is still a problem in data loss in the RoI pooling approach that makes this architecture poor when it comes to instance segmentation. Mask R-CNN on the other hand, is very similar to Faster R-CNN but overcomes this challenge by having the RoI align layer instead of RoI pooling. Instead of quantizing the boundaries of each region like pooling, RoI align computes exact floating-point values, ensuring that the extracted features are perfectly aligned with the original input. It does better than pooling as it allows to preserve spatial pixel-to-pixel alignment for every region of interest and there is no information lost or quantization. A key addition to Mask R-CNN is the Feature Pyramid Network (FPN) [34]. FPN leverages the natural multi-scale pyramidal hierarchy present in deep CNNs. It is used to address the challenge of multi-scale feature representation. FPN consists of a bottom-up and a top-down pathway as shown in Figure A.9.

To process an input image in a sequential manner, Mask R-CNN architecture uses many key components as shown in Figure A.10. In addition to what is explained above about the two stages, at its last step, classification and bounding box regression are performed. The RoI aligned features are passed through a shared fully connected network (FCN), FCN predicts the class probabilities for each proposed region and regresses the coordinates of the bounding boxes to refine their positions and sizes. Finally, the mask head network is responsible for generating pixel-level masks for each detected object region [60].

Mask R-CNN is selected in this study because of its ability to result in very high-quality masks and its ability to perform well in this specific task of crack detection.

– **YOLO**

You Only Look Once (YOLO) represents the family of single-stage detectors. It is designed for speed and efficiency by treating instance segmentation as a single, unified problem. It looks at the whole image at once, divides it into a grid, and at once predicts bounding boxes, class probabilities, and segmentation masks in a single pass [61].

YOLOv8 was released in 2023, it leverages the strengths of previous versions but at the same time it follows the standard structure of a single-stage object detector, having three parts: a backbone, a neck, and a head.

a. Backbone

– The backbone is a CNN that is doing the feature extraction. It is responsible for taking the input image and converting it into a set of feature maps at various scales.

b. Neck

– The neck functions as an intermediate layer connecting the backbone to the head. It aims to aggregate the feature maps generated by the backbone. YOLOv8 integrates a Path Aggregation Network (PANet) with a Feature Pyramid Network (FPN).

- FPN to create a top-down path that passes high-level semantic features to lower layers.
- PANet to add a bottom-up path that passes low-level positional features to deeper layers.

c. Head

– As the last component, the head is responsible for producing the output predictions.

- A key feature of the Yolov8 head is its anchor-free design. Unlike the earlier versions that depend on predefined anchor boxes, it predicts the object's center point directly. This approach makes the training faster. The head operates on each feature map from the neck to perform object detection of different sizes.

YOLOv8 is chosen to be studied in this research as it represents the state-of-the-art model from the YOLO family in balancing speed and accuracy.

Chapter Four
Methodology

This chapter presents the dataset in detail, it also presents an overview of the specific implementations of each approach developed, focusing on the evaluation models used for automated crack detection.

4.1 Dataset

The dataset was created by collecting images from historical masonry structures in Nablus, Palestine. It includes a range of diverse crack patterns and structural conditions. The research began with an initial dataset designed for the statistical-based model and the machine learning based model. Dataset 1 consists of 251 images for which the ground truth was established by determining the optimal threshold value for each image manually. Even after expanding this dataset, as discussed in the following paragraph, it remained impractical for civil engineers to manually define a threshold value for each individual training image. The size of this set was constrained by the time consuming nature of the annotation process.

Deep learning models required more data, due to its data-hungry nature. Therefore, the dataset was expanded. resulting in a total of 2794 patches (each image has the size of 640×640) for dataset 2. Dataset 2 name is NADER-Crack, where "Nader - نادر" means rare in Arabic, reflecting the unique characteristics of the dataset, and also honors my late husband, Nader who assisted in data collection. The suffix "Crack" highlights the dataset focus on crack detection.

The NADER-Crack dataset is annotated with three different classes: Brick, broken brick and crack. The first one is the 'Brick' class, it includes all the sound and intact masonry units in the structure that form the foundation of the walls. The second class is the 'Broken_Brick' class, it captures bricks that show signs of damage and degradation. The 'Crack' class is the most important one, it represents the narrow gaps that appear along the joints between the bricks. All of these three classes allow for a more detailed understanding of the structures to support accurate instance segmentation.

The creation of the ground truth data was performed by the domain experts using Roboflow [62], a platform that was selected for the annotation task for its comprehensive

suite of tools that optimize the workflow. Roboflow's interface is user friendly and easy to be used by non technical engineers. It made the polygons drawing around the visible classes efficient and fast. The annotated dataset used in this study was also hosted on Roboflow under the project titled Crack Detection. It can be accessed at: https://app.roboflow.com/crackdetection-1fwh1/crack-thesis/5

The unique characteristics of the dataset are reflected in Figure A.12, it shows samples from the dataset and Figure A.13 shows them after manual labelling. Unlike flat asphalt or smooth concrete, the masonry surfaces in NADER-Crack dataset have high intra-class texture variance, the background can easily be confused with a crack. The uniqueness in this case comes from forcing the models to learn the differences between texture noise and structural fractures. The dataset also captures cracks at different distances allowing the models to detect cracks from a distance or up close. Finally, actual fragmentation of the stone unit itself is present in the dataset and labelled by broken_brick where stone failure can also be captured. Table B.2 presents a comparative analysis between NADER-Crack dataset and benchmark datasets in the field of crack detection. Unlike other datasets, NADER-Crack is specifically designed to address the challenges of historical masonry structures by introducing multi-class instance segmentation, enabling precise crack detection. As noticed in the table, various dataset sizes are presented. The required dataset size depends on the model. In the case of transfer learning for example, fewer images are needed compared to training from scratch. Many studies show that fine-tuning pretrained models can achieve strong performance with several hundred annotated samples per class. For example, in paper [63], only 118 images from the CFD dataset were used to train and validate the Mask R-CNN model for the crack detection task and achieved good results.

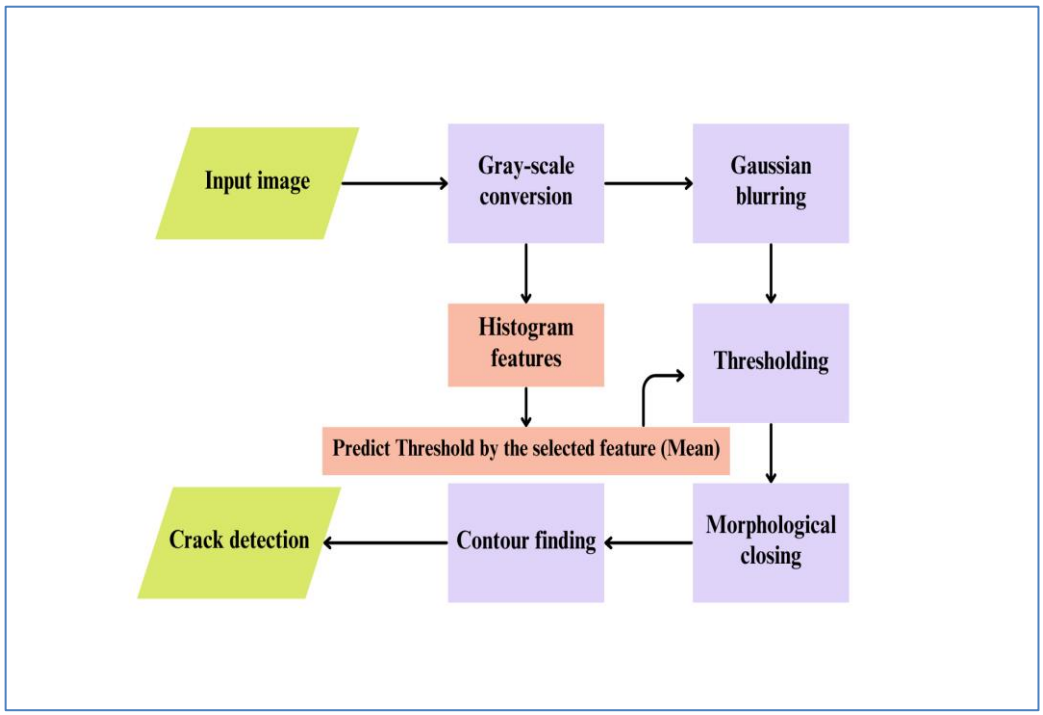
4.2 First Approach: Statistical-Based Crack Detection Model

This section outlines the development of a multi-stage model for automated crack detection. The images are processed through a series of algorithmic image processing steps, including thresholding, morphological operations, and feature extraction, to isolate potential regions of interest (RoI). As shown in the flowchart in Figure 2. The model operates through a sequential workflow, beginning with image pre-processing by gray-scale conversion and gaussian blurring, followed by the application of the proposed smart

dynamic thresholding and then by the image post-processing step where morphological closing is used for contour detection.

Figure 2

Flowchart of the proposed statistical-based crack detection model.



4.2.1 Image Pre-Processing

Crack detection logic relies on intensity differences, not color information. Based on this, a gray-scale conversion step, along with gaussian blurring, is applied first to reduce the computational complexity of image processing techniques while focusing on the most important features for subsequent processing. In the gaussian blurring step, a (3×3) kernel is applied to the gray-scale images to reduce noise and smooth out stone textures.

4.2.2 Smart Dynamic Thresholding (SDT)

To reduce the amount of available data on an image from pixels to regions, and to segment the RoI (cracks) from the background, a binary thresholding technique is applied. In static thresholding, a single threshold value (T) is chosen for the whole image, pixels with intensity values greater than this threshold are classified as foreground, and those with intensity values lower than the threshold are classified as background.

The value of T is determined based on analysis of some example images with the help of the domain expert. The domain expert specified the best T values so that the cracks can be isolated and identified. Initial experiments with a single, static threshold value for all images proved ineffective, as it failed to adapt to the significant variations in lighting and contrast present across the dataset. Therefore, smart dynamic thresholding (SDT) technique is proposed and T value is calculated for each individual image based on its characteristics. This technique provides an automatic recognition system without any intervention from the domain expert, and no manual work will be needed for larger datasets.

It has been observed that the thresholding value T varies between images according to their appearance, which is affected mainly by the brightness, color and texture in the image. Hence, seven histogram features are extracted from the images to identify the general appearance and to study their relation with the value of T. They include the minimum, maximum, mean, standard deviation, skewness energy and entropy of the gray-scale image.

To evaluate the effectiveness of histogram based features in crack detection, these seven different features are analyzed against threshold values defined by the structural domain expert. In Table 1, these histogram features are explained with their denotations in detail.

An initial Exploratory Data Analysis (EDA) is performed on this dataset. It shows a clear and a complete dataset. The target variable (Expert threshold) is slightly right-skewed. Its mean = 78.58, while the median = 70. Figure A.31 presents a comprehensive analysis of the seven input features. For each feature, a histogram is shown on the left to illustrate the data's frequency distribution and shape, and on the right box plot is shown to summarize the statistical properties and to identify outliers. Some features are symmetric like the mean, while others are left or right skewed, with many points clustered at the extremes. This indicates that some pre-processing techniques might be needed when using specific types of algorithms.

Since the relationship between some features and the expert-defined thresholds appeared non-linear, an empirical analysis was conducted to determine the relationship between the input histogram features and the expert-defined threshold values. A comparative regression study was performed using five mathematical models: linear, logarithmic,

power series, exponential and polynomial. As demonstrated in Table B.3, the polynomial model was selected as the final baseline as it yielded to the highest R^2 value.

The selection process is presented in Figure A.32. It shows the fit of the various regression curves against the mean value, confirming that the polynomial approach was the best empirical choice. For each feature, a polynomial function is applied and the coefficient of determination is calculated to assess the goodness of each fit [64]. R^2 values provided a quantitative measure of how well each feature correlated with the expert's criteria. In Figure A.14, the seven histogram features are compared using polynomial regression models. Each graph shows the relationship between a specific histogram feature (x-axis) and the expert-defined thresholds (y-axis). The red line is the fitted polynomial mode. The R^2 value indicates the goodness of this fit and is shown for each model in the following Table.

Table 1

R^2 value of polynomial regression model for each histogram feature

Feature	Min	Max	Mean	STD	Skewness	Energy	Entropy
R^2	0.65	0.49	0.77	0.51	0.44	0.57	0.56

The 'Mean' feature is selected for predicting T value due to its higher R^2 value. It is considered to be the most representative feature. To guide the thresholding process, mean is used to find T based on the following equation:

$$\mathrm{T} = 0.0273\,\mathrm{m}^2 - 7.15\,\mathrm{m} + 522 \qquad \text{(Eqn. 6)}$$

Where m is the mean value of the gray levels in the image. This formula automates the expert's decision-making process, making the system robust and adaptive.

4.2.3 Image Post-Processing

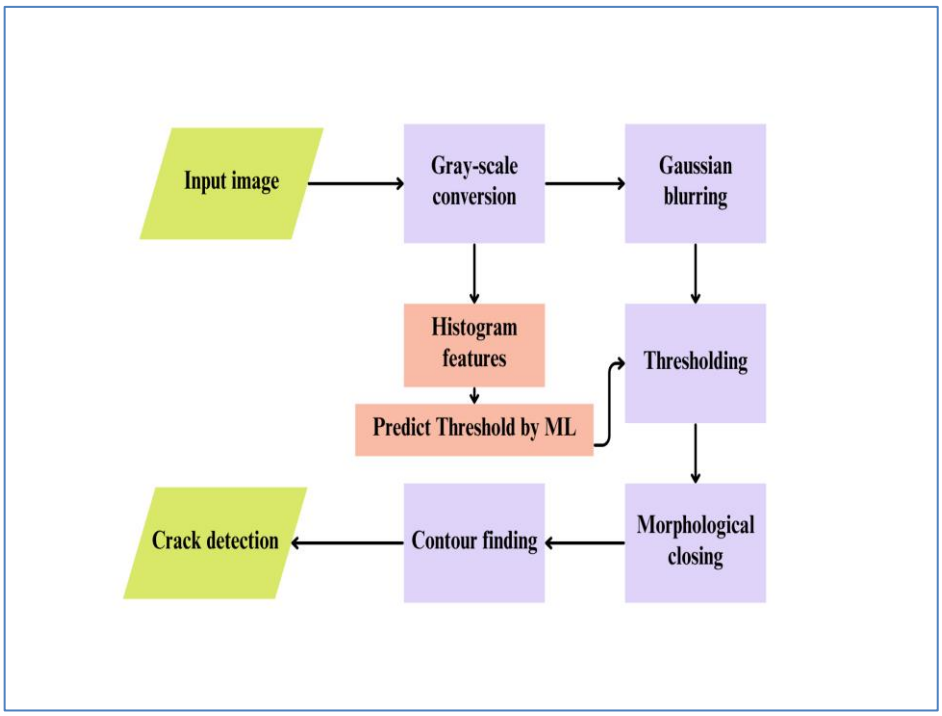
The detected cracks after the segmentation step appear disconnected and have holes. To fix this problem, morphological image processing is used to reduce the noise, fill the detected regions and connect close objects. Closing with an elliptical structuring element with a kernel size of (35×35) is applied to modify the shape of the objects in the resulting binary images.

4.3 Second Approach: Machine Learning-Based Crack Detection Model

This section details the development of the ML-based model, a more advanced approach designed to overcome the limitations of the statistical-based model in the first approach. The main innovation of this hybrid method is the integration of a machine learning regressor to optimize the most critical step of the model: image thresholding. A trained ML model analyzes statistical and textural features extracted from each image, to predict an optimal threshold value, instead of just depending on a fixed mathematical function. This predicted value is fed into the other stages of the IP model. This way ML capabilities are combined with the structured computer vision flow.

Figure 3

Flowchart of the proposed ML-based crack detection model



4.3.1 Feature Engineering & Data Analysis

Feature engineering involves creating features and transforming raw data into features that improve the performance of machine learning models [65]. The feature engineering process focuses on feature extraction in this study. So, instead of depending on raw pixel values, a descriptive feature vector of 13 statistical and textural properties is computed for each image in the dataset created for this second ML-based approach. This vector is

designed to summarize the image's characteristics in a way that is actually relevant to the task of threshold prediction. The features are summarized in the table below.

Table 2

Description of extracted image features

Feature Type	Feature Name	Description
Statistical	min, max	Minimum and maximum pixel intensity in the image.
	mean	The average pixel intensity, indicating overall brightness.
	Std_dev (STD)	Standard deviation of pixel intensities, measuring contrast.
	skew, kurtosis	Measures of the asymmetry and tailedness of the pixel histogram.
Textural	contrast	The local intensity variation between a pixel and its neighbors.
	dissimilarity	A measure of how different the elements of the GLCM are.
	homogeneity	Measures the closeness of the distribution of elements in the GLCM.
	energy	The sum of squared elements in the GLCM (also known as uniformity).
	entropy	A statistical measure of randomness in the image texture.
	correlation	A measure of how correlated a pixel is to its neighbors.
	ASM	Angular Second Moment, another measure of uniformity.

An explicit feature selection step is unnecessary, as the selected ML models inherently perform implicit feature selection during training by assigning higher importance to more predictive features.

Following feature extraction, an Exploratory Data Analysis (EDA) is performed on this dataset. The main goal of the exploratory data analysis is to understand the essential characteristics and relationships within the data to guide the model selection process [66]. The analysis focused on three key areas.

1. Target variable distribution

To study the distribution of the threshold values (our target variable), a histogram is plotted as shown in Figure A.15. Values show a wide distribution, ranging from low to high, with some focus in the 50 to 90 range. This confirms that a static threshold value for all images would be ineffective for this dataset. It also tells that a dynamic, predictive model is needed. No extreme outliers are being presented, which is going to be good when the model is implemented.

2. Correlation analysis

To understand which features are the most important and predictive, the correlation between the input features and the target variable (threshold value) is analyzed using a correlation matrix as shown in Figure A.16. The matrix highlights both positive and negative correlations with darker colors indicating stronger positive correlations. Mean, energy, homogeneity and ASM features show the strongest positive correlations with the target. This suggests that these features are highly related to the threshold value. On the other hand, some features like contrast and dissimilarity show negative correlations. In conclusion, no single feature has a strong enough correlation to be used as a one predictor, as what has been proposed in the first approach. Hence, developing a model capable of interpreting feature combinations is essential.

3. Relationship analysis (features-to-target)

Scatter plots are generated to visually explore the relationship between each independent variable and the target. Each of the 13 engineered features is plotted against the threshold value. This analysis reveals that none of the features showed a linear relationship with the

target variable. This finding is the primary justification for selecting advanced ML models.

The EDA shows that different features have different ranges as shown in Figure A.17, therefore, a crucial pre-processing step is needed which is feature scaling [67]. Without applying scaling, ML algorithms that are more likely to be sensitive to the magnitude of features might incorrectly assign more importance to features with larger numerical ranges, leading to poor performance. A standard scaler is used to transform each feature by subtracting its mean and dividing it by its standard deviation as shown in the equation below, where x is the original value, μ is the mean value, and σ is the standard deviation [68]. By doing this, each feature is rescaled to have a mean of 0 and a standard deviation of 1 enabling all features to contribute in a uniform way, which enhances the stability and reliability of model training.

$$z = \frac{x - \mu}{\sigma} \quad \text{(Eqn. 7)}$$

4.3.2 Model Selection

Based on the findings from the EDA, it is noticed that simple linear models will be insufficient for this dataset. Relationships between the image features and the target threshold value are complex and need non-linear regression ML models. Three ML models are tested to ensure a comprehensive and diverse evaluation.

1. Random forest regressor

As mentioned in the foundational concepts section above, a random forest regressor is resistant to overfitting and is a good baseline model for the task of finding the optimal threshold value.

2. XGBoost regressor

To test the gradient boosting model, XGBoost is selected. It is known that this model has a good performance when it comes to structured, tabular data like the feature set in this study.

3. Support vector regressor (SVR)

In addition to being effective at capturing nonlinearity in patterns, the SVR model foundation requires testing since it uses a different approach than the two models above.

– **Training & Hyperparameter Tuning**

Training is conducted using the dataset 1 containing 135 images for training and 116 images for testing. Each image is represented by a vector of 13 features standardized by standard scaler technique as mentioned in the pre-processing section above. Each of the three proposed ML methods is trained on these features to learn the mapping between the input image features and the target expert threshold value.

To find the best ML model from the three models, with the model configurations that yielded the best possible predictive performance for this specific crack detection task, a hyperparameter tuning process is conducted for each method and then evaluated. The tuning methodology used is called: grid search with cross-validation [69], [70]. It explores a predefined set of hyperparameter values for a given model, and for each combination, it evaluates the model's performance using cross-validation. Cross-validation repeatedly partitions the dataset into separate training and testing sets. Grid Search with 5-fold cross-validation is used. The training data is split into 5 "folds." For each parameter combination, the model is trained 5 times, each time using 4 of the folds for training and the remaining fold for validation. The average performance across all 5 folds is taken as the final score for that parameter set. The metric used to evaluate the performance of each parameter combination within the grid search is "negative mean absolute error," effectively optimizing the models to minimize the MAE.

The specific parameter grids explored for each of the three regression models are as presented in Table 3.

Table 3

Description of ML models and its hyperparameters

Model	Hyperparameter	Description	Values Searched
Random Forest	n_estimators	Number of trees in the forest.	[100, 200]
	max_depth	Max depth of each tree.	[10, 20, None]
	min_samples_split	Minimum samples to split a node.	[2, 5]
XGBoost	n_estimators	Number of boosting rounds (trees).	[100, 200]
	learning_rate	Step size shrinkage for each round.	[0.05, 0.1]
	max_depth	Maximum depth of each tree.	[3, 5, 7]
SVR	C	Regularization parameter.	[10, 100, 1000]
	gamma	Kernel coefficient for ‘rbf’.	[‘scale’, ‘auto’]

4.4 Third Approach: Transfer Learning Based Crack Detection Model

Transfer learning (TL) was crucially needed in this study. While the original small dataset used in approach 1 and 2 was expanded from 251 to 2794 for deep learning experiments, it is still considered relatively small for training complex models like Mask R-CNN and YOLOv8 from scratch by the end-to-end approach. Transfer learning enabled the use of state-of-the-art deep learning architectures like Mask R-CNN and YOLOv8 without requiring a huge dataset, thereby making the deep learning approach highly effective for this task.

- **Data Augmentation**

Data augmentation stands as an image processing technique which expands training dataset size through generating versions of existing images [71]. The process of applying transformers like rotations, flips with color adjustments produces new training data which stops model overfitting while enhancing model performance on new data points [72]. The initial NADER-Crack dataset is expanded by data augmentation. The training set size is

tripled from 1160 to 2794 images, ensuring that the model learns from a wider variety of data, making it able to generalize to new, unseen images that may have different orientations or lighting conditions. The applied transformations can be categorized into two groups: geometric and color-space augmentations as shown in Table 4.

Table 4

Data augmentation process

Data Augmentation Category	Transformations	Description
Geometric Augmentations	Flip	Images were randomly flipped both horizontally and vertically. This teaches the model that cracks are still cracks, regardless of their orientation.
	Rotation	Each image was randomly rotated by a small angle, between -15° and +15°. This helps the model become invariant to minor changes in camera angle.
	Crop	A random section of each image was cropped, zooming in by up to 27%. This forces the model to learn to identify cracks from partial views and at different scales.
	Blur	A slight Gaussian blur of up to 1.2 pixels was applied to simulate images that may be slightly out of focus.
	Noise	Random noise was added to up to 0.1% of pixels in an image, mimicking sensor noise from a digital camera.
Color-Space Augmentations	Saturation	The color intensity was randomly adjusted between -25% and +25%.
	Brightness	The overall brightness of the image was randomly modified between -15% and +15%.
	Exposure	The exposure was randomly altered between -10% and +10%.

- **Model Architecture**

Two state-of-the-art deep learning architectures with different fundamentals are selected to provide a comprehensive benchmark, Mask R-CNN represents the two-stage detection

framework, whereas YOLOv8 is employed for the single-stage approach. These 2 architectures are configured to train dataset 2, which was partitioned using a standard 7:2:1 split ratio for training, validation, and testing.

– **Mask R-CNN**

Mask R-CNN architecture in this study is built by using the same architecture explored in the original paper [59]. The main component is the ResNet-50 network, used as the backbone, followed by an FPN as the neck. ResNet-50 is a 50-layer deep convolutional neural network, developed by Microsoft Research [73]. ResNet is an acronym for Residual Network, these networks are developed to tackle the problems in training very deep neural networks especially the problem of vanishing gradients where deeper models perform worse than shallower ones [74]. The ResNet-50 network is considered as a valuable reference point in many studies and it is known that it gives excellent gains in both accuracy and speed. The architecture is described in Figure A.18.

To enable transfer learning for this work's particular case and dataset, weights pre-trained on the COCO dataset are used to initialize the model.

– **YOLO**

YOLOv8 model, as mentioned in the foundational concepts section above, is well known for its balance of speed and accuracy. Unlike two-stage models, YOLOv8 treats instance segmentation as a single regression problem, performing all tasks in one pass. This makes it efficient for real-world or real-time applications. The YOLOv8m-seg variant is chosen for this study. Compared to lighter variants like (YOLOv8n-seg or YOLOv8s-seg), YOLOv8m-seg offers improved detection precision, at the same time, it remains less computationally expensive than larger variants like YOLOv8l-seg. This makes it a perfect choice for crack detection.

In YOLOv8 architecture, the module used throughout the backbone is the C2f as shown in Figure A.19. It consists of two convolutional layers and bottlenecks. The bottleneck is a residual module that also consists of two convolutions. The Neck of the YOLOv8 architecture is a Path Aggregation Network (PANet). It is responsible for aggregating the features generated by the backbone.

To enable transfer learning for this work's particular case and dataset, weights pre-trained on the COCO dataset are used to initialize the model.

4.5 Performance Evaluation Metrics

The evaluation methodology in this research employs a unified evaluation protocol. It is important to ensure a fair comparison between the three different approaches. That's why the same consistent test set of 116 test images is used for the final evaluation of the three methodologies. For each of the 116 images, the final output of each pipeline generates a set of predicted crack contours. Then, a comparison against the ground-truth annotations for the "crack" class is conducted. This unified protocol provides an unbiased foundation for the comparative analysis presented in the results chapter. The test set was held out and used only for final performance evaluation to ensure unbiased estimation. This evaluation strategy is widely used in computer vision tasks where transfer learning is applied. Although cross validation or repeated runs have a chance of providing additional robustness by reporting average metrics, such approaches were not adopted in our work due to the significant computational cost associated with training the model. Given the moderate dataset size and the use of transfer learning, the held-out validation protocol represents a practical evaluation approach.

The deep learning models are trained to detect three classes (crack, brick, and broken_brick), but for the purpose of comparative analysis, only the performance on the "crack" class is considered.

To evaluate the performance at different stages of this research, a set of evaluation metrics is used. Some metrics are used to evaluate the regression models in approach 2, others are applied to approach 3, where deep learning is implemented, and finally, to conduct a comparative analysis across all proposed approaches. The method behind each one of these metrics are explained in this section.

– **Mean Absolute Error (MAE)**

In this work, MAE is used for regression model evaluation in approach 2. It is used to measure the average absolute difference between the model's predicted threshold and the expert-defined ground-truth threshold based on the following formula [75].

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \quad |y_i - \hat{y}_i| \qquad (Eqn. 8)$$

– **Root Mean Squared Error (RMSE)**

Similar to MAE, RMSE measures the average prediction error. But by having the square as shown in its equation, RMSE gives a much higher weight to large errors [75].

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \quad \text{(Eqn. 9)}$$

– **Intersection over Union (IoU)**

To conduct a fair comparison, per contour evaluation is performed. This is based on the concept of IoU, also known as the Jaccard index [76].

$$\mathrm{IoU} = \frac{TP}{TP+FP+FN} \quad \text{(Eqn. 10)}$$

Each detected contour is classified as follows:

– True Positives (TP): Predicted contour where the model correctly identifies a class.
– False Positives (FP): Predicted contour where the model incorrectly flags regions as ground-truth contour.
– False Negatives (FN): A ground-truth contour that was not detected by the model.

Logically, IoU measures the quality of the predicted cracks against the annotated ground truth cracks. To calculate it, binary mask generation is done first for both predicted and ground truth contours. If the pixel is inside the 'crack' class it will be considered as '1', otherwise it will be a '0'. Then, intersection is found by looking at where the number of pixels for ground truth masks and the predicted masks = '1'. Union is found by looking where either of these masks = '1'. The ratio of these two gives us the final IoU result.

Following the calculation of the IoU for each of the possible contour pairs, the evaluation methodology used in this paper applies a specific IoU threshold (defined with the civil engineer), to classify each prediction. If the IoU of a prediction with its ground truth meets or exceeds the threshold, it is classified as a true positive. If it was less than the required threshold, the detection is rejected and considered as a false positive. To find the false negatives, any ground-truth contour that was not claimed by a successful true positive prediction are classified as false negatives.

– **Precision**

Precision measures the accuracy of the positive predictions [77]. If precision = 40%, it means that for every 10 predictions the model flags as a crack, only 4 are actually correct. Precision in general measures reliability, but in our case getting a high precision is not the main goal. Our main goal is achieving the highest possible safety. So, even if a crack is detected with low precision, it flags an area of deterioration that requires attention.

$$Precision = \frac{TP}{TP+FP} \qquad (Eqn.\ 11)$$

– **Recall**

Measures the model's ability to find all the actual classes. If recall = 40%, it means that out of the 10 real cracks present, the model found only 4 and missed 6 potential safety hazards, which is really dangerous in our case. Achieving a high recall is our main goal in this study. It measures safety and sensitivity.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad (\text{Eqn. } 12)$$

– **F1-Score**

The F1-Score is the harmonic mean of precision and recall. It tries to balance the trade-off between false positives and false negatives [78]. The following equation shows how it is calculated.

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \qquad (\text{Eqn. } 13)$$

The F1-Score is selected as a fundamental metric for its ability to be used fairly, comparing the diverse methodologies, from statistical to deep learning methods.

– **Detection Rate**

The final evaluation metric is designed in collaboration with a civil engineer to provide critical risk assessment that goes beyond the F1-Score. This expert-driven detection rate is considered as the safety benchmark. The civil engineer determined that the ultimate priority is to minimize the false negatives (missed cracks). It is similar to the Recall equation, but the detection rate metric counts any successful instance overlap as a 'hit', where IoU is more than 0. Detection rate is calculated as the ratio of successfully detected ground truth instances to the total number of ground truth instances across the entire test set.

This metric is selected as the main one in assessing the model's maximum sensitivity and providing the necessary data to quantify the safety margin. This way, even if a crack is detected with a low precision, it still means that it needs attention from the civil engineer and it might be flagged as a risk.

Chapter Five
Experimental Design & Setup

5.1 Experimental Setup

The implementation of this project was programmed using Python 3. The selection of the libraries used was critical for each stage. GPUs were utilized to accelerate the demanding computational training and fine-tuning process. The main software packages used are detailed in Table 5 below.

Table 5

Software libraries used in this research

Library	Purpose
OpenCV	The primary library for all classical computer vision and image processing tasks, including color conversion, blurring, thresholding, and contour detection.
Pandas	Used for all data manipulation, including loading and structuring the feature datasets from Excel and CSV files.
NumPy	The fundamental package for numerical computation, used for handling image arrays and mathematical operations.
Scikit-learn	The core framework for the classical machine learning approach, providing implementations for Random Forest, SVR, data scaling (StandardScaler), and hyperparameter tuning (GridSearchCV).
XGBoost	Used for the implementation of the high performance Extreme Gradient Boosting model.
Detectron2	A Facebook AI Research library used as the framework for implementing and training the Mask R-CNN model.
Ultralytics YOLO	The official framework used for implementing and fine-tuning the YOLOv8 model.
Matplotlib & Seaborn	Used for all data visualization, including generating the plots for the Exploratory Data Analysis (EDA) and the final result comparisons.

5.2 Model Training & Configuration

This section details the final, optimized configurations for each of the three tested methodologies. To get the best possible results, a hyperparameter tuning process was conducted in each approach.

– **Statistical-Based Crack Detection Model**

The statistical-based crack detection model is configured with fixed parameters, such as kernel sizes. These parameters are determined through preliminary experimentation. The final configuration for this model is presented in the table below.

Table 6

Final hyperparameter configuration for the statistical-based model

Stage	Parameter	Value	Values Searched
1. Gaussian Blurring	Kernel Size	(3×3)	(3×3) - (35×35)
2. Thresholding	Method	Custom formula	-
3. Morphological Closing	Kernel Shape	Elliptical	-
	Kernel Size	(35×35)	(3×3) - (35×35)

– **ML-Based Crack Detection Model**

The ML-based model is optimized via an exhaustive grid search with cross-validation to find the best ML model from the three suggested models and to identify the final model hyperparameters that yielded the best possible performance for this specific crack detection task.

First, for each model, the final configuration is selected based on the accuracy results measured by MAE and RMSE. The final best configurations are as follows:

Table 7

Final ML models hyperparameters

Model	Hyperparameter	Optimal Value
Random Forest	n_estimators	200
	max_depth	10
	min_samples_split	2
XGBoost	n_estimators	100
	learning_rate	0.05
	max_depth	3
SVR	C	100
	gamma	‘auto’

XGBoost was selected as the final optimal model for predicting threshold due to its highest accuracy based on the results of MAE and RMSE as shown in the table below.

Table 8

Accuracy results for ML models

Model	MAE	RMSE
Random Forest	14.99	18.40
XGBoost	14.92	18.37
SVR	15.48	19.01

The XGBoost model achieved an MAE of 14.92, meaning that, on average, its threshold prediction was only about 15 pixel values away from the expert-defined ground truth as shown in Figure A.20.

– **Transfer Learning**

For the Mask R-CNN R50-FPN selected architecture, transfer learning is leveraged by initializing the model with weights pre-trained on the COCO dataset. This will enable the model to build on top of an existing knowledge, and to reduce the training requirements compared to training the model from scratch.

The fine-tuning process is configured based on the best practices for this architecture using Kaggle GPUs which provides NVIDIA Tesla T4 with 16 GB of VRAM. Mask R-CNN has high memory requirements and at the same time there are some memory limitations that lead to some final configurations.

The batch size in our code is restricted to 2 to prevent out-of-memory errors and following what is noted in Detectron2 documentation. A base learning rate of 0.00025 is chosen as this is the standard, recommended value for fine-tuning Mask R-CNN models with a small batch size (like 2) in the Detectron2 framework. It is known to provide stable convergence. For maximum iterations, to ensure the model is trained for a duration comparable to the YOLOv8 experiments and to allow it sufficient time to learn the complex features of cracks, the total number of training iterations is set to 46000. Initial experiments with shorter training runs failed to achieve adequate model convergence for the difficult “crack” class.

“ROI Heads Per Image” parameter is the number of top-scoring candidate regions (RoI) from stage 1 that are passed to stage 2 for detailed analysis. This parameter is kept at its standard value = 128, to provide a good balance between speed and performance.

To train the three classes Brick, Broken_brick, and Crack, with these selected hyperparameters, approximately 5 hours of computation time was needed for each configuration.

The YOLOv8m-seg architecture accomplishes transfer learning by initializing the model from the YOLOv8m-seg.pt weights file, which contains parameters pre-trained on the COCO dataset. The fine-tuning process is configured for 100 epochs which also took around 5 hours. The training duration is empirically chosen based on validation metrics that showed performance convergence and stability without evidence of overfitting till the 100 epochs. A batch size of 16 is used to ensure stable gradients without exceeding the available GPU memory.

Table 9

Final Deep learning models hyperparameters

Model	Hyperparameter	Optimal Value
Mask R-CNN	Pre-trained Weights	COCO Dataset
	Images Per Batch	2
	Learning Rate	0.00025
	Max Iterations	46000
	ROI Heads Per Image	128
YOLOv8	Pre-trained Weights	COCO Dataset
	Epochs	100
	Batch Size	16

Chapter Six
Results & Discussion

This chapter presents the quantitative and qualitative analysis of the three crack detection approaches implemented in this work. First, the statistical-based model will be evaluated. Second, the hybrid approach, where ML is introduced to the same pipeline implemented in the first approach. And finally, transfer learning methods for both Mask R-CNN and YOLOv8 will be evaluated. After all of this, a final comparative analysis will be conducted. This chapter will also present the results of each approach visually using 9 images from the golden test set as shown in Figure A.21 with their corresponding annotations in Figure A.22. The first three subsections in this chapter show results when IoU threshold = 10%. This value is selected to be the first benchmark. In the comparative analysis section, another threshold value for the IoU will be discussed along with the detection rate.

6.1 Statistical-Based Crack Detection Model Performance

The performance of the statistical-based crack detection model is evaluated in this section with a step-by-step visual analysis of the image processing techniques used in this model. The test set of 116 unseen images is used to evaluate the process.

The same 9 test samples are being shown in Figure A.23 after applying the SDT thresholding technique with the same configurations and formulas explained in chapter 5. Closing results are also shown in Figure A.24. The final results are then presented in Figure 4. Predictions obtained from this approach are presented in red, ground truth annotations are shown in green, and the overlap between these two are in yellow. The visual results of these images show that this model archives a fair detection performance. However, it has failed to detect a large number of true cracks. For example, in the central image (e), the model failed to detect any crack. This particular image has varying lighting conditions, which caused the model to perform poorly in adapting to different scenarios. At the same time, in the first image and others, many of the predicted cracks are false positives. All of these limitations have their roots from the formula used to determine threshold values. Better results are expected when using machine learning to predict the thresholds. On the other hand, as discussed with the civil engineer expert, a good number of these false positives are considered to be abnormalities that may need attention. An

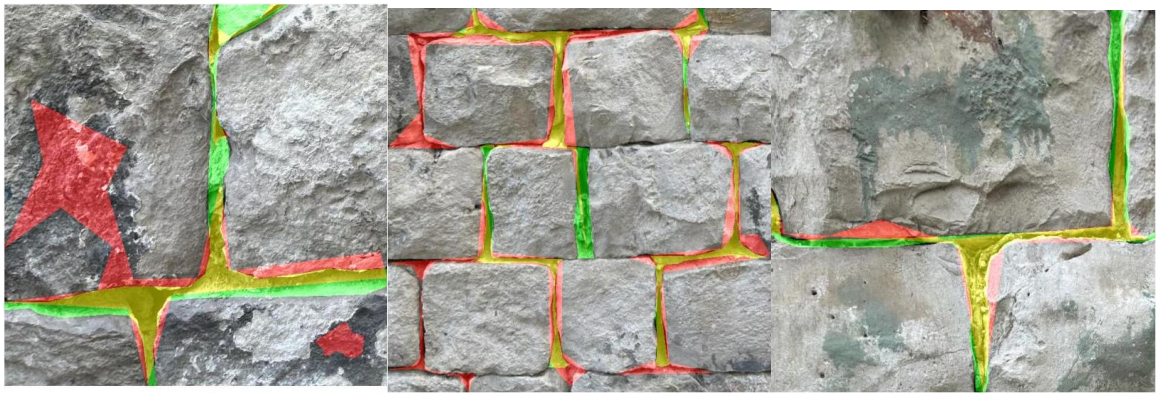
example of this is the first image (a) where the red area on the top left is discoloring of stone, which could be classified as a defect but not part of the study interest.

For quantitative assessment, the results were consistent with the visual observations. An F1-Score of 0.4054 is established as the performance baseline for this study. The analysis also reveals a significant challenge with this method, which is the low Recall of 0.3659, indicating that the statistical approach failed to detect a large number of true cracks and has a high number of false negatives. On the other hand, the Precision of 0.4545 is considered to be fair and modest.

In general, these results suggest that the suggested statistical method with the SDT thresholding is a reasonable start point, with many limitations causing the model to be non-adaptive to varying image conditions. This motivated a more dynamic, data-driven approach using machine learning.

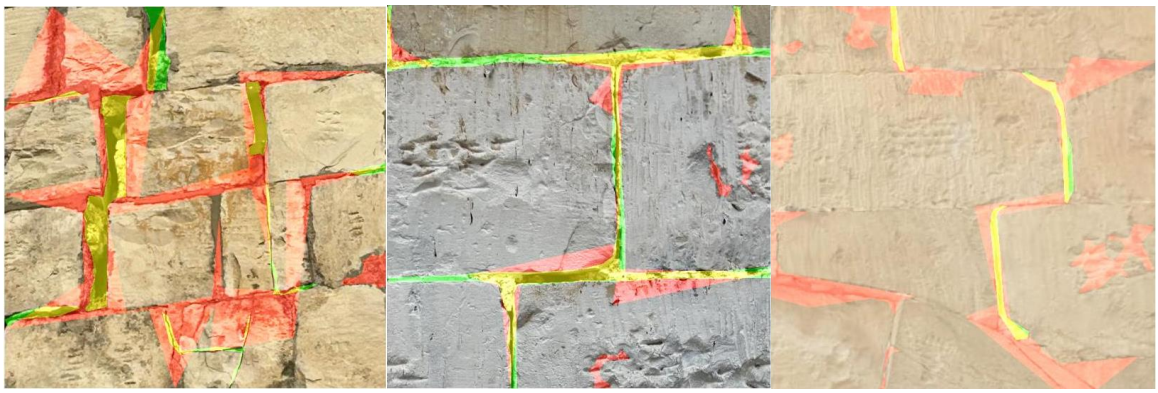
Figure 4

Final statistical-based crack detection results: Prediction (red), ground truth (green), and overlap (yellow) on 9 test samples



(a) (b) (c)

(d) (e) (f)



(g) (h) (i)

6.2 ML-Based Crack Detection Model Performance

In this section, the performance of the IP crack detection model with machine learning is evaluated through a step-by-step visual analysis of the image processing techniques it employs. The evaluation uses a test set of 116 previously unseen images.

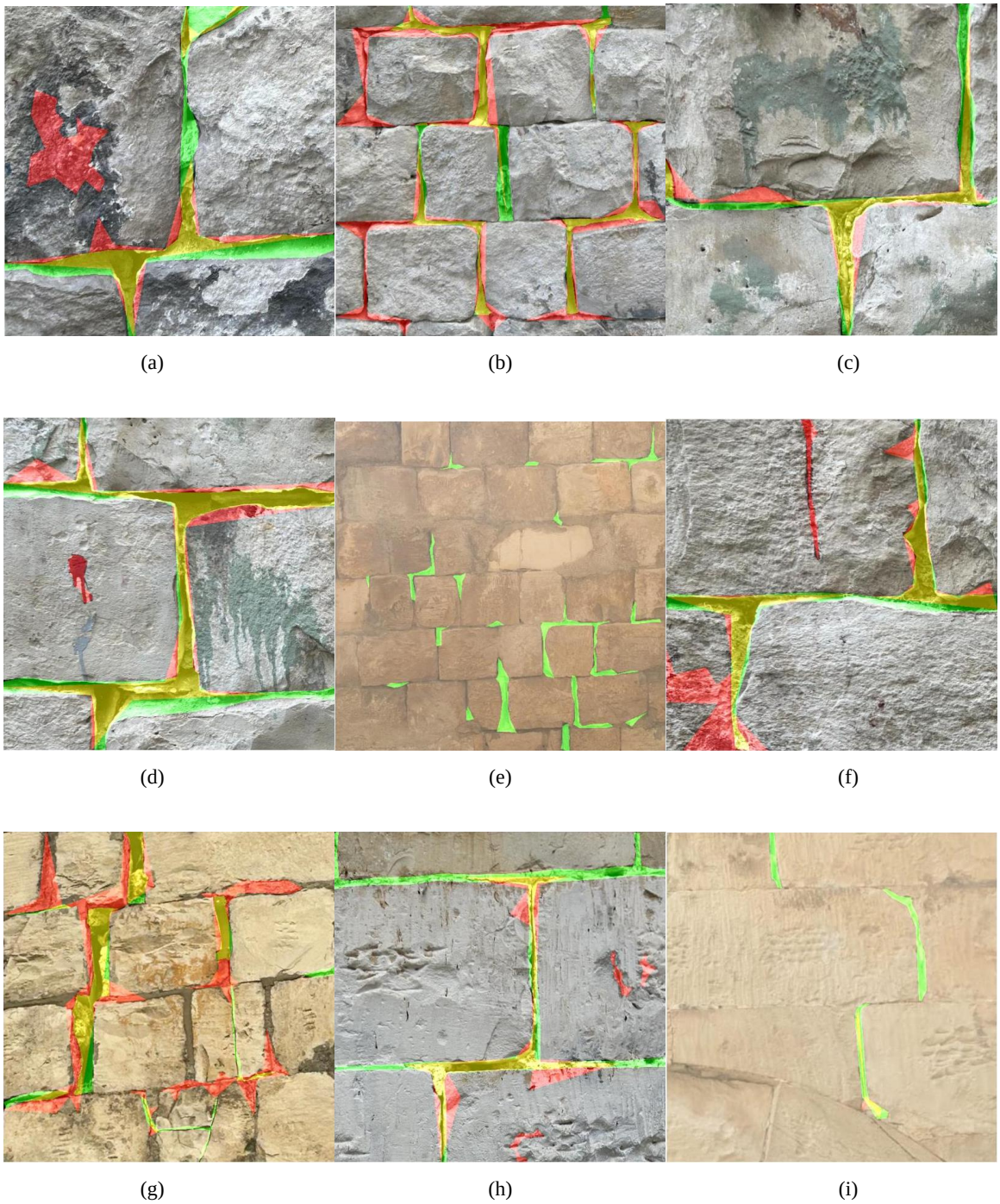
To compare the performance of the IP and the ML-based models, the visual step-by-step results of the hybrid model are also provided in Figure A.25 & Figure A.26. As seen in the first image (a) and in image (g) & (i) in Figure 5, the red regions indicating false predictions show a slight decrease. The hybrid model is better at reducing the number of false negatives. Still, the model failed to detect any crack in the central image (e).

For quantitative assessment, the results were also consistent with the visual observations in the second approach. With an F1-Score of 0.4483, the model outperforms the first approach by 10.6% points. The main driver of this improvement was the increase in Recall (from 0.3659 to 0.4228). On the other hand, the Precision of 0.4771 is better than the previous approach, but only slightly. The integration of a machine learning regressor to optimize image thresholding using more features was the main reason behind this improvement.

Although the ML-based model achieves better results, it still shares the same limitations introduced by the first approach. ML thresholding results outperforms SDT, morphological closing still exists in the second approach to connect cracks. However, its use can also cause some noise to merge and be falsely detected as cracks.

Figure 5

Final ML-based crack detection results: Prediction (red), ground truth (green), overlap (yellow) on 9 test samples



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

6.3 Transfer Learning Based Crack Detection Models Performance

To train Mask R-CNN & YOLOv8, Nader-Crack dataset was annotated with 3 different classes as mentioned in the previous sections and with the mentioned configurations. Each model was trained twice: once with three classes (Brick, Broken_Brick, and Crack)

and once with a single class (Crack). This setup was designed to evaluate whether including additional contextual classes would improve crack detection performance compared to training solely on crack annotations for each model.

Before evaluating the final model performance for both architectures on the test set, it is crucial to analyze the training process itself first. This analysis justifies the selected hyperparameters mentioned in the experimental setup chapter again here, and confirms that the models learned effectively from the training data and did not suffer from overfitting.

Overfitting is the main challenge in training, it occurs when a model starts to memorize the training data, including noise instead of actually learning the underlying hidden patterns. An overfit model will perform well on the data it has seen but will fail on new, unseen data.

The first method used to avoid overfitting is data augmentation, mentioned in detail in chapter 4. Data augmentation made the model learn the hidden features of a crack, rather than memorizing the specific orientation or lighting of the training images, thereby improving its ability to generalize to new, unseen data.

The second method is having the validation set periodically checked throughout the training process. The validation loss was the key metric monitored, with the technique of early stopping. Early stopping involves stopping the training process as soon as the validation loss stops improving.

A key feature of the Ultralytics framework is its built-in validation monitoring. By default, at the end of each training epoch, the framework automatically evaluates the model's performance on the validation dataset. It then logs key metrics, such as validation loss. The framework also performs best model checkpointing. It continuously tracks the primary validation metric and saves a copy of the model's weights only when that metric improves. Because YOLOv8 experiments were conducted using the official Ultralytics framework, its training was conducted first. The validation loss curves were easily observed and were found to converge and stabilize around 100 epochs, indicating that this duration was sufficient for the model to reach its peak performance without overfitting. Then, to create a controlled fair comparison between the two architectures, both the YOLOv8 and Mask R-CNN models were trained for the same duration of 100

epochs. This ensured that each model was exposed to the training data an identical number of times.

As shown in Figure A.27, the training loss for both the Mask R-CNN and YOLOv8 models in different training experiments consistently decreased and began to plateau, which is the classic indicator of successful model convergence.

Starting with Mask R-CNN model evaluation on multi-class configuration, the model achieved an F1-Score of 0.3211. While it demonstrated a very high Recall of 0.7358, indicating it was sensitive to identify most ground-truth cracks, unfortunately, this was coupled with an extremely low Precision of 0.2053.

In order to improve performance, the model was re-trained with a single-class detector. It was unfortunate that the F1-Score dropped to 0.2970. The model's Recall increased slightly to 0.8028 but its Precision worsened to 0.1822, indicating a higher rate of false positives. This indicates that the Mask R-CNN architecture benefited from the contextual information present in the multi-class environment. The multi-class model was better and able to learn what was not a crack.

This may have occurred due to the multi-stage architecture of Mask R-CNN. As mentioned in the foundational concepts chapter, in stage 1 (proposal), the model identifies RoI without considering their content, and in stage 2 (classification), it examines each region to assign a class. This two-step architecture can lead to poor performance when only a single class is used, as the model has less contextual information to inform its decisions in these two stages.

For YOLOv8 multi-class configuration, the model achieved an F1-Score of 0.5432, the YOLOv8 model established itself as a very high-performing baseline compared to all other models. An excellent balance is noticed between the high Recall of 0.4919 & the Precision of 0.6065. Meaning that the YOLOv8 model was reliable in its predictions and also effective at detecting a significant portion of the ground-truth cracks.

To evaluate the potential for further performance optimization, the model was re-trained with a single class. Unlike Mask R-CNN, results became better. F1-Score was boosted to 0.5730. Recall stayed stable, but Precision increased to 0.6068. Figure A.28 shows this comparison.

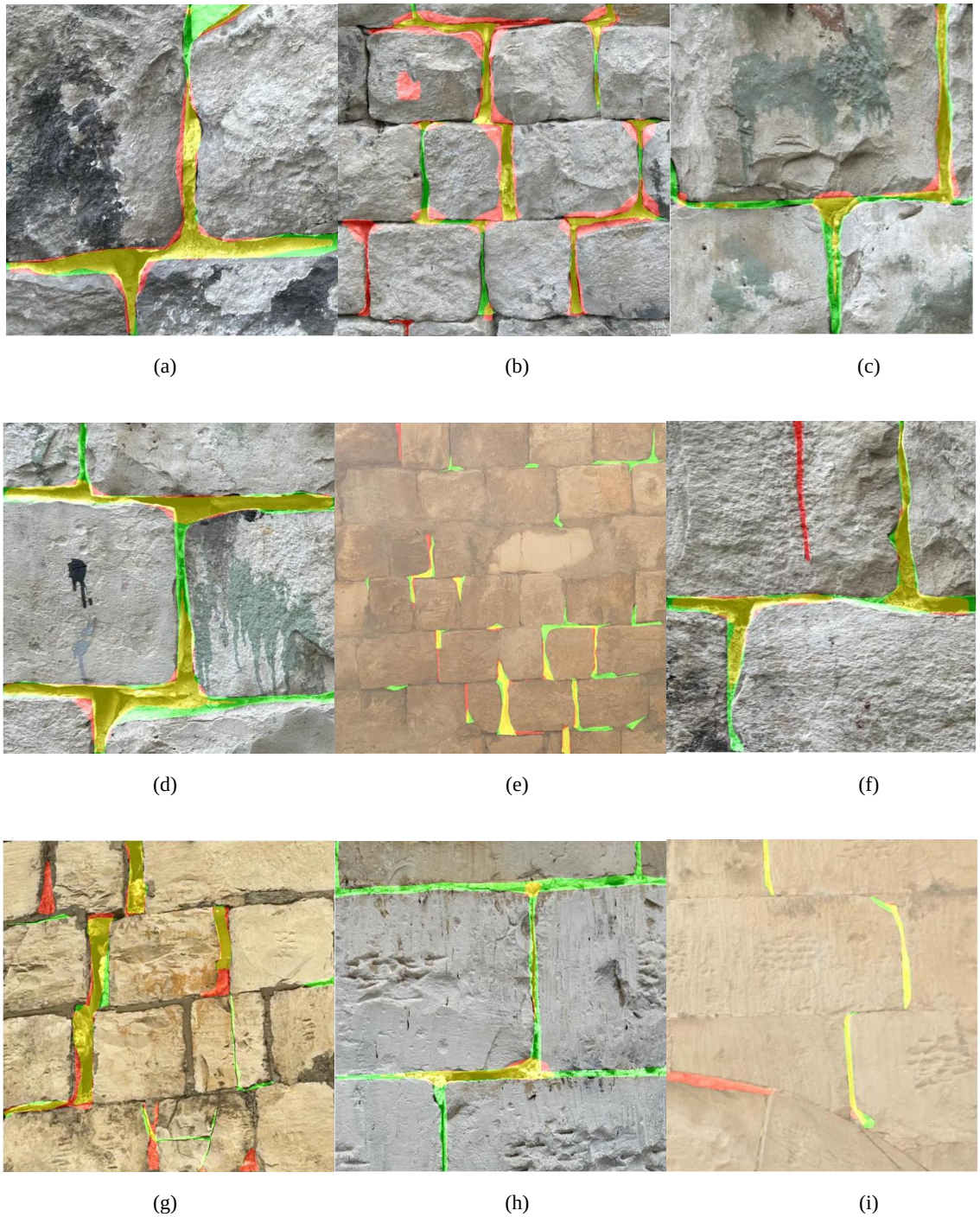
We expect that this increase in performance happened due to YOLOv8 single-stage architecture and its ability to analyze the entire image context at once with its attention completely focused on one class. The model focused entirely on learning crack features.

A main clear difference of the deep learning approach is its end-to-end nature. Unlike the multi-stage statistical-based and the ML-based approaches, it does not produce explicit, intermediate results such as a thresholded or morphologically closed image. Instead, the neural network learns to map the raw input pixels directly and without any manual work needed to the final segmentation masks. The visual results for the deep learning models, as will be shown in the following figures, describe the final predicted masks on top of the original test images.

It is worth mentioning that if both models were applied to a different dataset featuring a different stone type, the transfer learning approach would enable adoption to the new geometries through another short fine tuning phase, potentially requiring only a few hundred new samples. This shows the high scalability and generalizability of transfer learning methods compared to rule based approaches.

Figure 6

Final crack detection results by Mask R-CNN trained on multi-class annotations (Brick/ Broken_Brick/ Crack): Prediction (red), ground truth (green), overlap (yellow) on 9 test samples



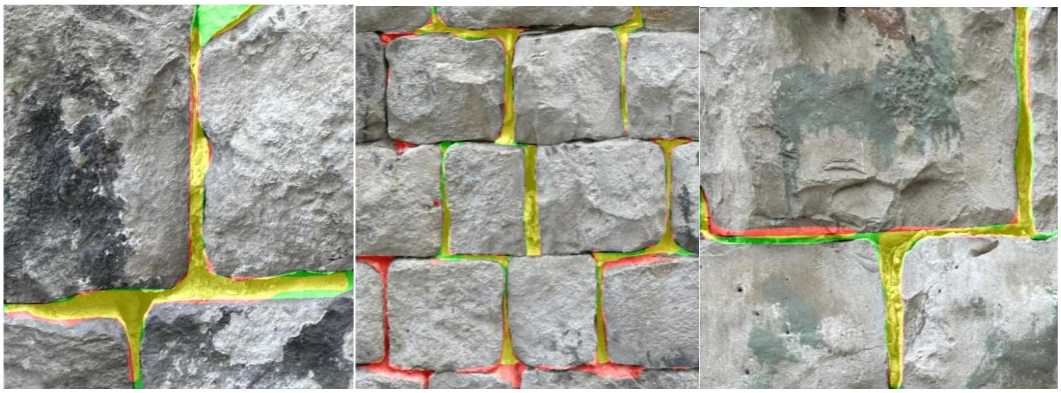
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 7

Final crack detection results by Mask R-CNN trained on single-class annotations (Crack): Prediction (red), ground truth (green), overlap (yellow) on 9 test samples



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 8

Final crack detection results by YOLOv8 trained on multi-class annotations (Brick/ Broken_Brick/ Crack): Prediction (red), ground truth (green), overlap (yellow) on 9 test samples

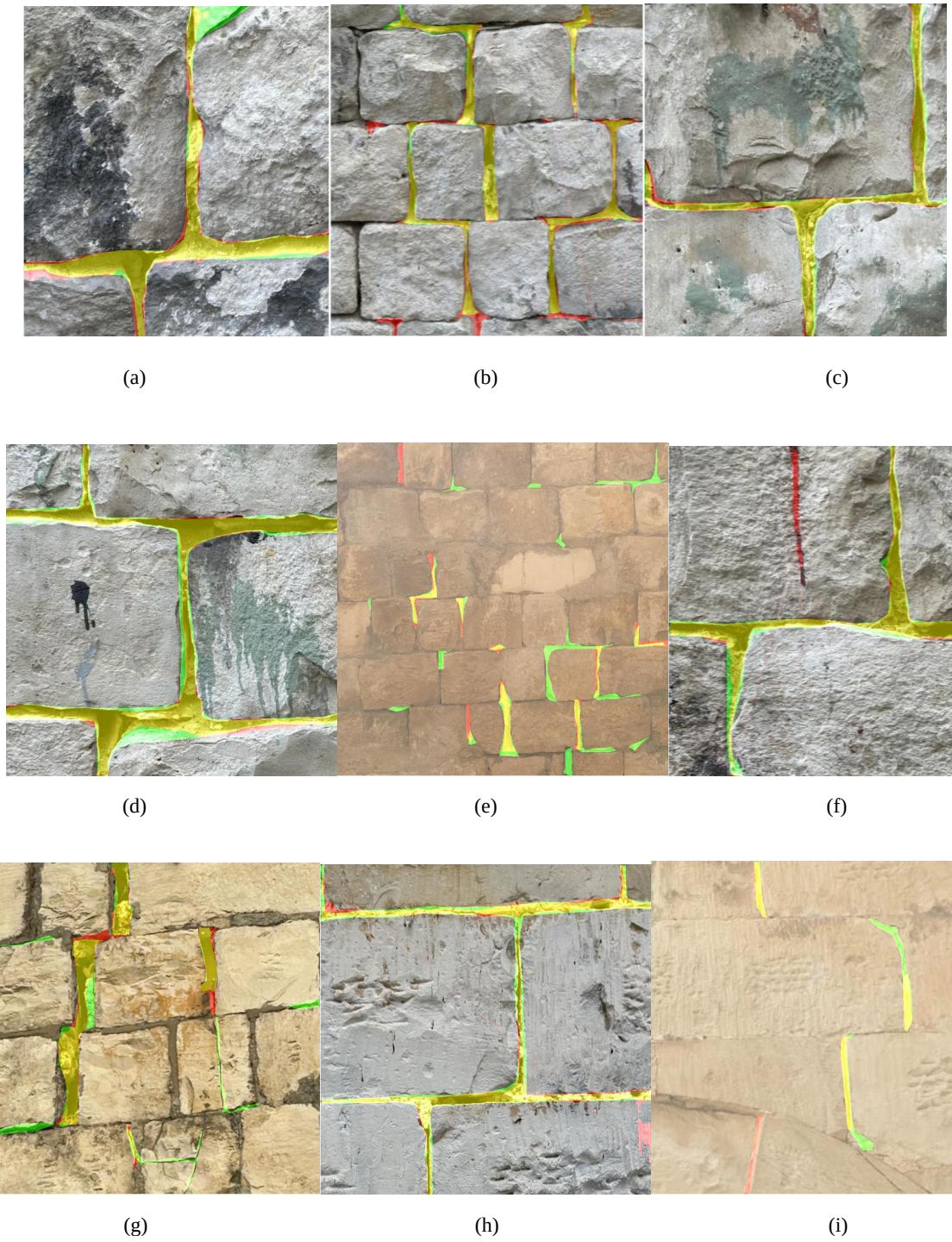
(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

Figure 9

Final crack detection results by YOLOv8 trained on single-class annotations (Crack): Prediction (red), ground truth (green), overlap (yellow) on 9 test samples



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

6.4 Comparative Analysis of Crack Detection Approaches

As explained in the methodology chapter, the evaluation applies a specific IoU threshold defined by the civil engineer to classify each detection and then calculate the F1-Score based on this threshold. The previously explained results are presented based on using

IoU = 0.10. The selection of this initial IoU was the direct result of a review session with the civil engineer expert. Unlike solid objects like bricks, cracks are thin and irregular, so it made sense to start with this IoU value. The expert then noted that the prediction could follow the path of a crack, but due to being a bit thinner or wider than the ground truth, it results in a low IoU. Although an IoU threshold of 0.50 is common and adopted in generic object detection benchmarks such as PASCAL VOC [79], these thresholds are used for well-defined objects with regular shapes. However, in our application the target structures are thin and irregular. This makes IoU sensitive to minor boundary deviations and small spatial shifts.

The primary goal of this work is to detect the presence of cracks for structural inspection. Accordingly, application-specific evaluation metrics were designed in collaboration with civil engineers to ensure relevance to practical requirements. Lower IoU thresholds were adopted to reflect real-world engineering acceptability.

The final results of each approach are listed in Table 10 (a) for IoU threshold = 0.10, showing a clear hierarchy of performance. The ML-based model outperforms the statical-based one. This confirms the hypothesis that a data-driven model is more accurate and robust than the formula for predicting the optimal threshold value.

Mask R-CNN achieved the highest recall by a significant margin, outperforming the first two approaches. This indicates that the architecture of it was the most sensitive at identifying potential crack regions. However, this high recall cost a low precision. In contrast, the single-class YOLOv8 model showed a superior balance between precision and recall, leading to the highest overall F1-Score among all of these approaches.

A visual comparison was conducted on sample images from the test set to understand the practical differences in each model's behavior as shown in Figure 10. The ground-truth crack is shown in green, while each model's prediction is shown in red. Areas of overlap appear in yellow.

In the first image, it is clear that the first approach failed to detect a portion of the main crack (a false negative). At the same time, it identifies a large area on the left as a crack while it is not (a large false positive). The second hybrid model was a bit more accurate, resulting in a lower false positive (red area decreased). However, the underlying image processing pipeline steps like morphological closing still limits its precision. Deep

learning approaches gave very clear and better results. This figure provides powerful visual evidence that corroborates the quantitative results.

To comprehensively assess the models' performance, IoU was set to 0.01, keeping in mind that the primary objective is crack localization and existence rather than finding precise crack measurements. The IoU = 0.01 was selected after testing many thresholds, as the minimum overlap criterion. The impact of this threshold was efficient. Upon applying it, the performance metrics for all methodologies improved slightly as shown in Table 10 (b).

By looking at the detection rate, we found that Mask R-CNN achieved the highest score, demonstrating itself to be the most sensitive model. Mask R-CNN could identify 89.23 of all defects. It is considered to be the superior warning system. The reason for this success lies in its RPN that acts as a highly sensitive screening tool, it proposes a vast number of potential regions. In a scenario where a human inspector reviews every detection and the cost of a missed crack is really high, this model is considered to be the most effective. The classical methods achieved a higher detection rate than the YOLOv8 models due to their noisy nature of generating more noise that has a higher chane of overlapping with a ground truth crack. YoloV8 is still considered as a conservative model for this scenario.

Figure 10

The output of the primary approaches on the same sample image

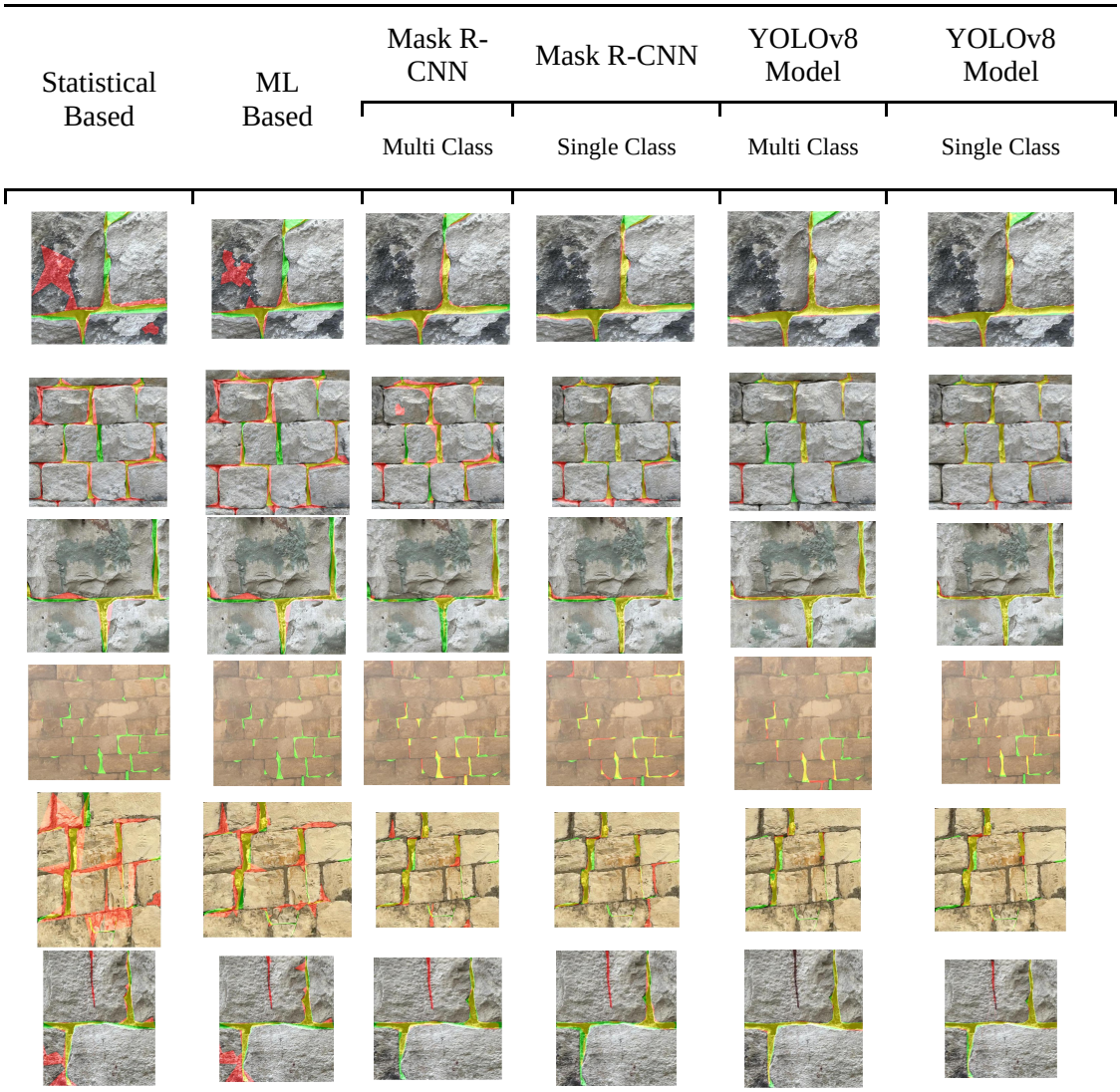


Table 10

Evaluation of crack detection approaches

Approach	Precision	Recall	F1-Score
Statistical-Based (IP)	0.4545	0.3659	0.4054
ML-Based (Hybrid)	0.4771	0.4228	0.4483
Mask R-CNN (Multi-class)	0.2053	0.7358	0.3211
Mask R-CNN (Single-class)	0.1822	0.8028	0.2970
YOLOv8 (Multi-class)	0.6065	0.4919	0.5432
YOLOv8 (Single-class)	0.6068	0.5427	0.5730

(a) IoU threshold = 0.10

Approach	Precision	Recall	F1-Score
Statistical-Based (IP)	0.5354	0.4309	0.4775
ML-Based (Hybrid)	0.5482	0.4858	0.5151
Mask R-CNN (Multi-class)	0.2223	0.7967	0.3477
Mask R-CNN (Single-class)	0.1946	0.8577	0.3173
YOLOv8 (Multi-class)	0.6291	0.5102	0.5634
YOLOv8 (Single-class)	0.6477	0.5793	0.6116

(b) IoU threshold = 0.01

Approach	Detection Rate
Statistical-Based (IP)	0.6850
ML-Based (Hybrid)	0.7114
Mask R-CNN (Multi-class)	0.8232
Mask R-CNN (Single-class)	**0.8923**
YOLOv8 (Multi-class)	0.5711
YOLOv8 (Single-class)	0.6789

(c) Detection Rate (Safety Margin/ Recall when IoU > 0)

Chapter Seven
Conclusion & Future Work

7.1 Conclusion

This thesis conducted a comparative analysis of three AI models for automated crack detection in Palestinian historical buildings: A statistical-based model, an ML-based model and deep learning models. The research confirmed that the role of data-driven learning enhances detection performance. The ML-based model F1-Scores outperformed the statistical-based model in different evaluation settings; this was a direct result of including more features in the analysis. However, the most important finding was the good accuracy achieved from deep learning approaches. The single-class Mask-RCNN model offered the highest safety coverage (Detection Rate: 89.2%) establishing it as the most accurate and reliable solution. On the other hand, the single-class YOLOv8 model is still considered a good model in balancing precision and recall (F1-Score: 61.16%).

7.2 Future Work

Although this research offers a thorough comparison, acknowledging its limitations is important. Primarily, the data collection phase proved challenging, as it required capturing many images within different real-world environments. This labor-intensive and time-consuming process suggests that a truer degree of model robustness and generalizability would necessitate a more extensive dataset. Furthermore, the annotation stage alone spanned over two months.

The complexity of the dataset itself presents another obstacle, the regular occurrences of cracks alongside broken bricks made the task difficult. Additionally, the specific masonry style includes clear mortar lines that often mimic the appearance of cracks.

These findings open opportunities for future work. Future research is encouraged to utilize NADER-Crack dataset as a standard benchmark for automated crack detection. The next step would be to try more advanced deep learning architectures like YOLOv11.

List of Abbreviations

Abbreviation	Meaning
ASM	Angular Second Moment
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CV	Computer Vision
EDA	Exploratory Data Analysis
FN	False Negative
FP	False Positive
TP	True Positive
FPN	Feature Pyramid Network
GLCM	Gray-Level Co-occurrence Matrix
GPU	Graphics Processing Unit
IP	Image Processing
IoU	Intersection over Union
MAE	Mean Absolute Error
PANet	Path Aggregation Network
RMSE	Root Mean Squared Error
RoI	Region of Interest
RPN	Region Proposal Network
SVR	Support Vector Regression

References

[1] E. Mesquita, P. Antunes, F. Coelho, P. André, A. Arêde, and H. Varum, "Global overview on advances in structural health monitoring platforms," *J. Civ. Struct. Health Monit.*, vol. 6, no. 3, pp. 461–475, Jul. 2016, doi: 10.1007/s13349-016-0184-5.

[2] C. L. Blavier, H. E. Huerto-Cardenas, N. Aste, C. Pero, F. Leonforte, and S. della torre, "Adaptive measures for preserving heritage buildings in the face of climate change: A review," *Build. Environ.*, vol. 245, p. 110832, Sep. 2023, doi: 10.1016/j.buildenv.2023.110832.

[3] M. M. Salameh, B. A. Touqan, J. Awad, and M. M. Salameh, "Heritage conservation as a bridge to sustainability assessing thermal performance and the preservation of identity through heritage conservation in the Mediterranean city of Nablus," *Ain Shams Eng. J.*, vol. 13, no. 2, p. 101553, Mar. 2022, doi: 10.1016/j.asej.2021.07.007.

[4] R. Tresidder and E. L. Deakin, "Historic buildings and the creation of experiencescapes: looking to the past for future success," *J. Tour. Futur.*, vol. 5, no. 2, pp. 193–201, Jan. 2019, doi: 10.1108/JTF-04-2019-0034.

[5] A. Khalilia, "Effect of Architecture on the Cultural Perception of the Palestinian Society: The Example of Nablus," *Book"Heritage City Values Beyond,"* p. 75, 2022.

[6] U. W. H. Centre, "Old Town of Nablus and its environs," UNESCO World Heritage Centre. Accessed: Jun. 16, 2025. [Online]. Available: https://whc.unesco.org/en/tentativelists/5714/

[7] "Nablus, Enduring Heritage and Continuing Civilisation - The revitalization plan of the old cityCAUE du Nord." Accessed: Jun. 16, 2025. [Online]. Available: https://www.caue-nord.com/fr/portail/41/mediatheque/27097/nablus-enduring-heritage-and-continuing-civilisation-the-revitalization-plan-of-the-old-city.html

[8] "History of Nablus." Accessed: Jun. 16, 2025. [Online]. Available: http://www.shahincomputer.com/nablus/nablus/history.htm

[9] P. Sorrentino, G. Brandonisio, and A. De Luca, "Complex monumental buildings. Definition of complexities and structural implications," *Procedia Struct. Integr.*, vol. 44, pp. 1664–1671, Jan. 2023, doi: 10.1016/j.prostr.2023.01.213.

[10] H. Wang, Y. Shi, Q. Yuan, and M. Li, "Crack Detection and Feature Extraction of

Heritage Buildings via Point Clouds: A Case Study of Zhonghua Gate Castle in Nanjing," *Buildings*, vol. 14, no. 8, Art. no. 8, Aug. 2024, doi: 10.3390/buildings14082278.

[11] A. T. Council, *Seismic Performance Assessment of Buildings*. Federal Emergency Management Agency, 2012.

[12] H. Zhang, L. Ma, Z. Yuan, and H. Liu, "Enhanced concrete crack detection and proactive safety warning based on I-ST-UNet model," *Autom. Constr.*, vol. 166, p. 105612, Oct. 2024, doi: 10.1016/j.autcon.2024.105612.

[13] M. R. Jahanshahi, J. S. Kelly, S. F. Masri, and G. S. Sukhatme, "A survey and evaluation of promising approaches for automatic image-based defect detection of bridge structures," *Struct. Infrastruct. Eng.*, vol. 5, no. 6, pp. 455–486, Dec. 2009, doi: 10.1080/15732470801945930.

[14] M. Mishra, "Machine learning techniques for structural health monitoring of heritage buildings: A state-of-the-art review and case studies," *J. Cult. Herit.*, vol. 47, pp. 227–245, Jan. 2021, doi: 10.1016/j.culher.2020.09.005.

[15] H. Kaveh and R. Alhajj, "Recent advances in crack detection technologies for structures: a survey of 2022-2023 literature," *Front. Built Environ.*, vol. 10, p. 1321634, Jul. 2024, doi: 10.3389/fbuil.2024.1321634.

[16] D. Kang and Y. Cha, "Autonomous UAVs for Structural Health Monitoring Using Deep Learning and an Ultrasonic Beacon System with Geo-Tagging," *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 33, May 2018, doi: 10.1111/mice.12375.

[17] Y. Hamishebahar, H. Guan, S. So, and J. Jo, "A Comprehensive Review of Deep Learning-Based Crack Detection Approaches," *Appl. Sci.*, vol. 12, no. 3, Art. no. 3, Jan. 2022, doi: 10.3390/app12031374.

[18] R. E. Philip, A. D. Andrushia, A. Nammalvar, B. G. A. Gurupatham, and K. Roy, "A Comparative Study on Crack Detection in Concrete Walls Using Transfer Learning Techniques," *J. Compos. Sci.*, vol. 7, no. 4, Art. no. 4, Apr. 2023, doi: 10.3390/jcs7040169.

[19] A. Landstrom and M. J. Thurley, "Morphology-Based Crack Detection for Steel Slabs," *IEEE J. Sel. Top. Signal Process.*, vol. 6, no. 7, pp. 866–875, Nov. 2012, doi: 10.1109/JSTSP.2012.2212416.

[20] B. Y. Lee, Y. Y. Kim, S.-T. Yi, and J.-K. Kim, "Automated image processing technique for detecting and analysing concrete surface cracks," *Struct. Infrastruct. Eng.*, vol. 9, no. 6, pp. 567–577, Jun. 2013, doi: 10.1080/15732479.2011.593891.

[21] R. M. Haralick, S. R. Sternberg, and X. Zhuang, “Image Analysis Using Mathematical Morphology,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-9, no. 4, pp. 532–550, Jul. 1987, doi: 10.1109/TPAMI.1987.4767941.

[22] D. Loverdos, V. Sarhosis, E. Adamopoulos, and A. Drougkas, “An innovative image processing-based framework for the numerical modelling of cracked masonry structures,” *Autom. Constr.*, vol. 125, p. 103633, May 2021, doi: 10.1016/j.autcon.2021.103633.

[23] L. Truong-Hong and D. Laefer, “Micro vs. macro models for predicting building damage underground movements,” Jan. 2008.

[24] V. Sarhosis and J. V. Lemos, “A detailed micro-modelling approach for the structural analysis of masonry assemblages,” *Comput. Struct.*, vol. 206, pp. 66–81, Aug. 2018, doi: 10.1016/j.compstruc.2018.06.003.

[25] H.-G. Moon and J.-H. Kim, “Inteligent Crack Detecting Algorithm on the Concrete Crack Image Using Neural Network,” presented at the 28th International Symposium on Automation and Robotics in Construction, Seoul, Korea, Jun. 2011. doi: 10.22260/ISARC2011/0279.

[26] Y. Cha, W. Choi, and O. Büyüköztürk, “Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks,” *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, May 2017, doi: 10.1111/mice.12263.

[27] Y. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Buyukozturk, “Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types,” *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 00, pp. 1–17, Nov. 2017, doi: 10.1111/mice.12334.

[28] S. Dorafshan, R. Thomas, C. Coopmans, and M. Maguire, *Deep Learning Neural Networks for sUAS-Assisted Structural Inspections: Feasibility and Application*. 2018, p. 882. doi: 10.1109/ICUAS.2018.8453409.

[29] V. Sarhosis and D. Loverdos, “Automatic image-based brick segmentation and crack detection of masonry walls using machine learning,” *Autom. Constr.*, vol. 140, p. 104389, Aug. 2022, doi: 10.1016/j.autcon.2022.104389.

[30] D. Dais, İ. E. Bal, E. Smyrou, and V. Sarhosis, “Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning,” *Autom. Constr.*, vol. 125, p. 103606, May 2021, doi: 10.1016/j.autcon.2021.103606.

[31] K. Chaiyasarn, W. Khan, L. Ali, M. Sharma, D. Brackenbury, and M. DeJong,

Crack Detection in Masonry Structures using Convolutional Neural Networks and Support Vector Machines. 2018. doi: 10.22260/ISARC2018/0016.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 06, 2016, *arXiv*: arXiv:1506.01497. doi: 10.48550/arXiv.1506.01497.

[33] B. Marin, K. Brown, and M. S. Erden, "Automated Masonry crack detection with Faster R-CNN," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, Lyon, France: IEEE, Aug. 2021, pp. 333–340. doi: 10.1109/CASE49439.2021.9551683.

[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," Apr. 19, 2017, *arXiv*: arXiv:1612.03144. doi: 10.48550/arXiv.1612.03144.

[35] D. Kang, S. S. Benipal, D. L. Gopal, and Y.-J. Cha, "Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning," *Autom. Constr.*, vol. 118, p. 103291, Oct. 2020, doi: 10.1016/j.autcon.2020.103291.

[36] L. Ali, H. Jassmi, W. Khan, and F. Alnajjar, "Crack45K: Integration of Vision Transformer with Tubularity Flow Field (TuFF) and Sliding-Window Approach for Crack-Segmentation in Pavement Structures," *Buildings*, vol. 13, p. 55, Dec. 2022, doi: 10.3390/buildings13010055.

[37] R. Sapkota, D. Ahmed, and M. Karkee, "Comparing YOLOv8 and Mask RCNN for object segmentation in complex orchard environments," *Artif. Intell. Agric.*, vol. 13, pp. 84–99, Sep. 2024, doi: 10.1016/j.aiia.2024.07.001.

[38] Y. Choi, B. Bae, T. Hee Han, and J. Ahn, "Application of Mask R-CNN and YOLOv8 Algorithms for Concrete Crack Detection," *IEEE Access*, vol. 12, pp. 165314–165321, 2024, doi: 10.1109/ACCESS.2024.3469951.

[39] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, NY: Pearson, 2018.

[40] R. Szeliski, "Computer Vision: Algorithms and Applications, 2nd Edition".

[41] K. Kaur, D. Y., "Various image segmentation techniques: a review.," *Int. J. Comput. Sci. Mob. Comput.*, pp. 809–814.

[42] N. A. Rehman and F. Haroon, "Adaptive Gaussian and Double Thresholding for Contour Detection and Character Recognition of Two-Dimensional Area Using Computer Vision," *Eng. Proc.*, vol. 32, no. 1, Art. no. 1, May 2023, doi:

10.3390/engproc2023032023.

[43] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms”.

[44] D.-Y. Huang and C.-H. Wang, “Optimal multi-level thresholding using a two-stage Otsu optimization approach,” *Pattern Recognit. Lett.*, vol. 30, pp. 275–284, Jan. 2009, doi: 10.1016/j.patrec.2008.10.003.

[45] B. Gatos, I. Pratikakis, and S. J. Perantonis, “Adaptive degraded document image binarization,” *Pattern Recognit.*, vol. 39, no. 3, pp. 317–327, Mar. 2006, doi: 10.1016/j.patcog.2005.09.010.

[46] N. Neogi, D. K. Mohanta, and P. K. Dutta, “Defect Detection of Steel Surfaces with Global Adaptive Percentile Thresholding of Gradient Image,” *J. Inst. Eng. India Ser. B*, vol. 98, no. 6, pp. 557–565, Nov. 2017, doi: 10.1007/s40031-017-0296-2.

[47] D. Islam, T. Mahmud, and T. Chowdhury, “An efficient automated vehicle license plate recognition system under image processing,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 29, no. 2, p. 1055, Feb. 2023, doi: 10.11591/ijeecs.v29.i2.pp1055-1062.

[48] S. Kumar and V. Bhatnagar, “A Review of Regression Models in Machine Learning,” *J. Intell. Syst. Comput.*, vol. 3, no. 1, pp. 40–47, Dec. 2022, doi: 10.51682/jiscom.v3i1.30.

[49] L. M. Soegianto, A. T. Hinandra, P. A. Suri, and M. Fajar, “Comparison of Model Performance on Housing Business Using Linear Regression, Random Forest Regressor, SVR, and Neural Network,” *Procedia Comput. Sci.*, vol. 245, pp. 1139–1145, 2024, doi: 10.1016/j.procs.2024.10.343.

[50] H. A. Salman, A. Kalakech, and A. Steiti, “Random Forest Algorithm Overview,” *Babylon. J. Mach. Learn.*, vol. 2024, pp. 69–79, Dec. 2024, doi: 10.58496/BJML/2024/007.

[51] M. Awad and R. Khanna, “Support Vector Regression,” in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, M. Awad and R. Khanna, Eds., Berkeley, CA: Apress, 2015, pp. 67–80. doi: 10.1007/978-1-4302-5990-9_4.

[52] F. Zhang and L. J. O’Donnell, “Chapter 7 - Support vector regression,” in *Machine Learning*, A. Mechelli and S. Vieira, Eds., Academic Press, 2020, pp. 123–140. doi: 10.1016/B978-0-12-815739-8.00007-9.

[53] N. O. Mahony *et al.*, *Deep Learning vs. Traditional Computer Vision*, vol. 943. 2020. doi: 10.1007/978-3-030-17795-9.

[54] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” Feb. 21, 2015,

arXiv: arXiv:1405.0312. doi: 10.48550/arXiv.1405.0312.

[55] "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," *Ieee Trans. Med. Imaging*, vol. 35, no. 5, pp. 1285–1298, Feb. 2016, doi: 10.1109/TMI.2016.2528162.

[56] M. Wu *et al.*, "Prediction of the remaining time and time interval of pebbles in pebble bed HTGRs aided by CNN via DEM datasets," *Nucl. Eng. Technol.*, vol. 55, no. 1, pp. 339–352, Jan. 2023, doi: 10.1016/j.net.2022.09.019.

[57] "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects | IEEE Journals & Magazine | IEEE Xplore." Accessed: Sep. 27, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/9451544

[58] N. Bačanin Džakula, "Convolutional Neural Network Layers and Architectures," 2019, Accessed: Sep. 27, 2025. [Online]. Available: https://portal.sinteza.singidunum.ac.rs/paper/700

[59] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Jan. 24, 2018, *arXiv*: arXiv:1703.06870. doi: 10.48550/arXiv.1703.06870.

[60] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," Jan. 24, 2018, *arXiv*: arXiv:1703.06870. doi: 10.48550/arXiv.1703.06870.

[61] "What is YOLOv8? A Complete Guide." Accessed: Oct. 03, 2025. [Online]. Available: https://blog.roboflow.com/what-is-yolov8/

[62] *B. Dwyer, J. Nelson, T. Hansen et al., "Roboflow (Version 1.0) [Computer software]," 2025. [Online].* (2025). [Online]. Available: https://roboflow.com

[63] Y. Zhang, B. Chen, J. Wang, J. Li, and X. Sun, "APLCNet: Automatic Pixel-Level Crack Detection Network Based on Instance Segmentation," *IEEE Access*, vol. 8, pp. 199159–199170, 2020, doi: 10.1109/ACCESS.2020.3033661.

[64] G. B. Barrett, "The Coefficient of Determination: Understanding r squared and R squared," Mar. 2000, doi: 10.5951/MT.93.3.0230.

[65] C. Reid Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf, "A conceptual basis for feature engineering," *J. Syst. Softw.*, vol. 49, no. 1, pp. 3–15, Dec. 1999, doi: 10.1016/S0164-1212(99)00062-X.

[66] T. Kumar, M. Arora, V. Verma, S. Lalar, and S. Bhushan, "Pre-Examination of Breast Cancer Dataset Using Exploratory Data Analysis (EDA) Approach," in *2024 International Conference on Computational Intelligence and Computing Applications (ICCICA)*, Samalkha, India: IEEE, May 2024, pp. 1–7. doi:

10.1109/ICCICA60014.2024.10585026.

[67] J. M. H. Pinheiro *et al.*, “The Impact of Feature Scaling In Machine Learning: Effects on Regression and Classification Tasks,” Sep. 22, 2025, *arXiv*: arXiv:2506.08274. doi: 10.48550/arXiv.2506.08274.

[68] Z. L. Thakker and S. H. Buch, “Effect of Feature Scaling Pre-processing Techniques on Machine Learning Algorithms to Predict Particulate Matter Concentration for Gandhinagar, Gujarat, India,” *Int. J. Sci. Res. Sci. Technol.*, pp. 410–419, Feb. 2024, doi: 10.32628/IJSRST52411150.

[69] Y. Wu, Z. Zhang, X. Qi, W. Hu, and S. Si, “Prediction of flood sensitivity based on Logistic Regression, eXtreme Gradient Boosting, and Random Forest modeling methods,” *Water Sci. Technol.*, vol. 89, no. 10, pp. 2605–2624, May 2024, doi: 10.2166/wst.2024.146.

[70] T. Yan, S.-L. Shen, A. Zhou, and X. Chen, “Prediction of geological characteristics from shield operational parameters by integrating grid search and *K*-fold cross validation into stacking classification algorithm,” *J. Rock Mech. Geotech. Eng.*, vol. 14, no. 4, pp. 1292–1303, Aug. 2022, doi: 10.1016/j.jrmge.2022.03.002.

[71] K. Maharana, S. Mondal, and B. Nemade, “A review: Data pre-processing and data augmentation techniques,” *Glob. Transit. Proc.*, vol. 3, no. 1, pp. 91–99, Jun. 2022, doi: 10.1016/j.gltp.2022.04.020.

[72] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, and F. Shen, “Image Data Augmentation for Deep Learning: A Survey,” Nov. 05, 2023, *arXiv*: arXiv:2204.08610. doi: 10.48550/arXiv.2204.08610.

[73] “ResNet 50 | SpringerLink.” Accessed: Oct. 07, 2025. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4842-6168-2_6

[74] L. Borawar and R. Kaur, “ResNet: Solving Vanishing Gradient in Deep Networks,” 2023. doi: 10.1007/978-981-19-8825-7_21.

[75] T. O. Hodson, “GMD - Peer review - Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not”, Accessed: Oct. 07, 2025. [Online]. Available: https://gmd.copernicus.org/articles/15/5481/2022/gmd-15-5481-2022-discussion.html

[76] S. Wu, X. Li, and X. Wang, “IoU-aware Single-stage Object Detector for Accurate Localization,” Apr. 15, 2020, *arXiv*: arXiv:1912.05992. doi: 10.48550/arXiv.1912.05992.

[77] M. Zhu, “Recall, Precision and Average Precision”.

[78] D. M. W. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation,” Oct. 11, 2020, *arXiv*: arXiv:2010.16061. doi: 10.48550/arXiv.2010.16061.

[79] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal Visual Object Classes (VOC) Challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, Art. no. 2, Sep. 2009, doi: 10.1007/s11263-009-0275-4.

[80] “(PDF) The Support Vector Regression with the parameter tuning assisted by a differential evolution technique: Study of the critical velocity of a slurry flow in a pipeline,” *ResearchGate*, Aug. 2025, doi: 10.2298/CICEQ0803191L.

[81] “CNN in Deep Learning: Algorithm and Machine Learning Uses,” Simplilearn.com. Accessed: Oct. 03, 2025. [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network

[82] “【目标检测】Mask R-CNN - 知乎.” Accessed: Oct. 03, 2025. [Online]. Available: https://zhuanlan.zhihu.com/p/62492064?ref=blog.roboflow.com

[83] “Understanding ResNet50: A Comprehensive Guide to the Architecture,” thinkingstack. Accessed: Oct. 07, 2025. [Online]. Available: https://www.thinkingstack.ai/blog/business-use-cases-11/a-comprehensive-guide-to-resnet50-architecture-and-implementation-56

[84] “A High-Accuracy Deformable Model for Human Face Mask Detection | SpringerLink.” Accessed: Oct. 07, 2025. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-97-0376-0_8

[85] M. Maguire, S. Dorafshan, and R. Thomas, “SDNET2018: A concrete crack image dataset for machine learning applications,” *Browse Datasets*, May 2018, doi: https://doi.org/10.15142/T3TD19.

[86] “Automatic Road Crack Detection Using Random Structured Forests | IEEE Journals & Magazine | IEEE Xplore.” Accessed: Feb. 25, 2026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7471507

[87] F. Yang, L. Zhang, S. Yu, D. Prokhorov, X. Mei, and H. Ling, “Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection,” *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1525–1535, Apr. 2020, doi: 10.1109/TITS.2019.2910595.

[88] Y. Liu, J. Yao, X. Lu, R. Xie, and L. Li, “DeepCrack: A deep hierarchical feature learning architecture for crack segmentation,” *Neurocomputing*, vol. 338, pp. 139–

153, Apr. 2019, doi: 10.1016/j.neucom.2019.01.036.

[89] J. Valença, E. N. B. S. Júlio, and H. J. Araújo, “Applications of photogrammetry to structural assessment,” *Exp. Tech.*, vol. 36, no. 5, pp. 71–81, Dec. 2017.

Appendices

Appendix A

Figures

Figure A.1

Map of the Old City of Nablus showing key historical sites [7]

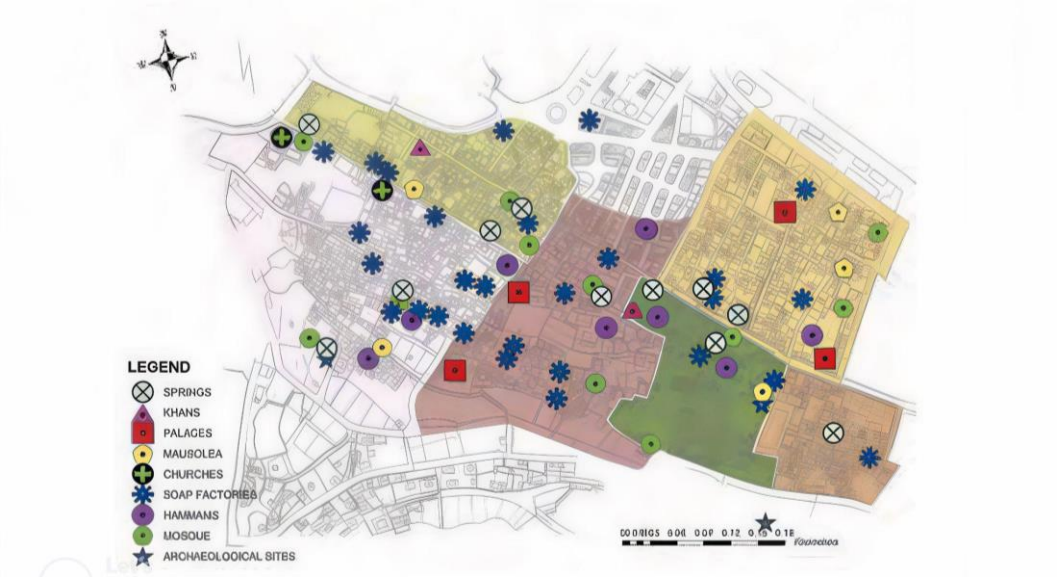


Figure A.2

Overview of crack detection approaches

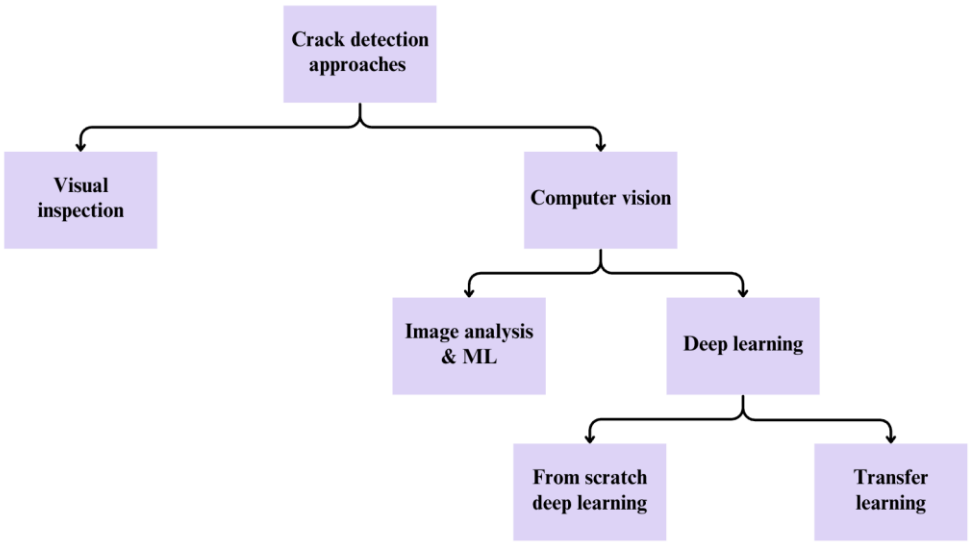


Figure A.3

(A) End-to-End DL model (B) Transfer learning model

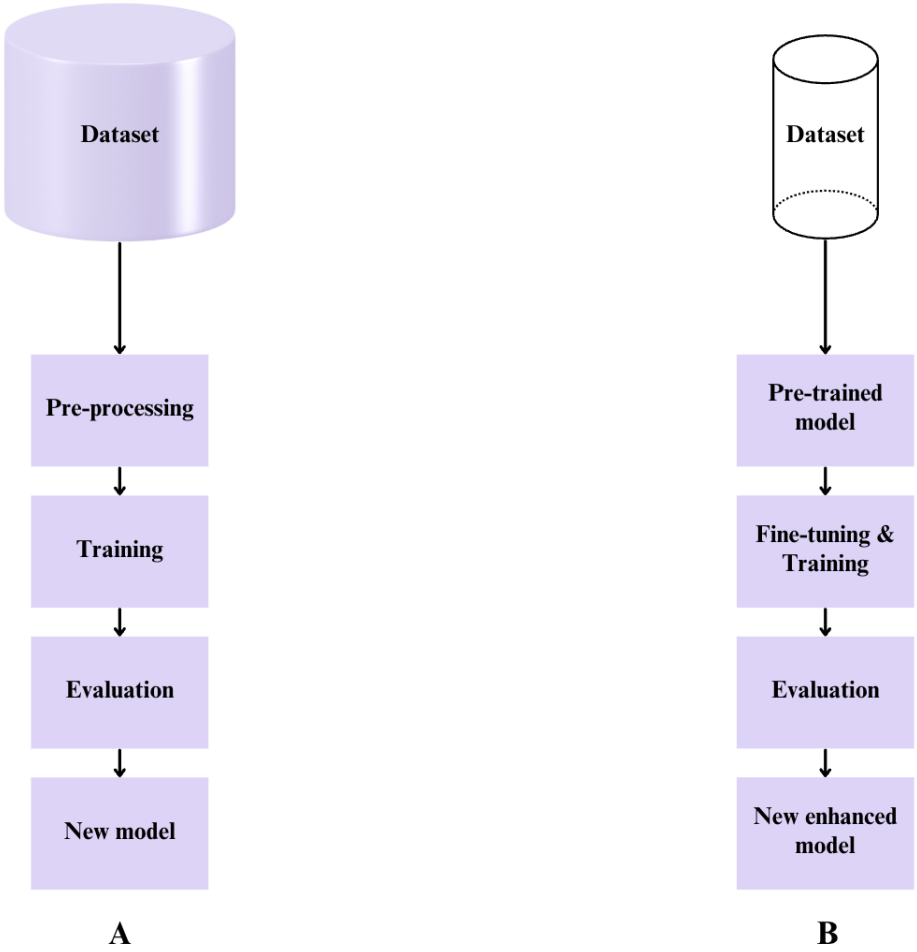


Figure A.4

Masonry modeling procedures: a) masonry sample; b) detailed micro modeling; c) simplified micro modeling; d) macro modeling [23]

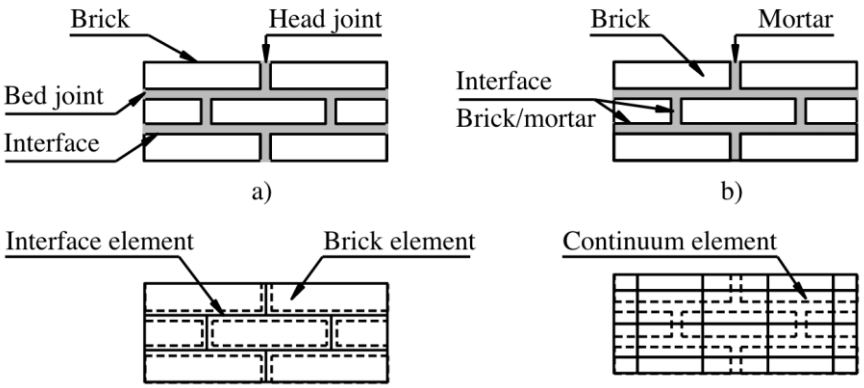


Figure A.5

Random Forest architecture

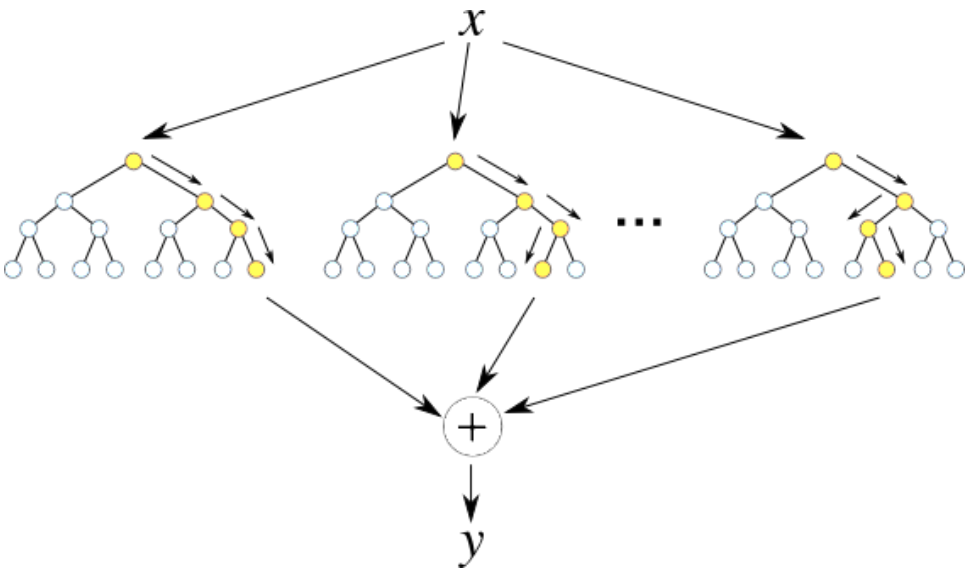


Figure A.6

XGBoost architecture

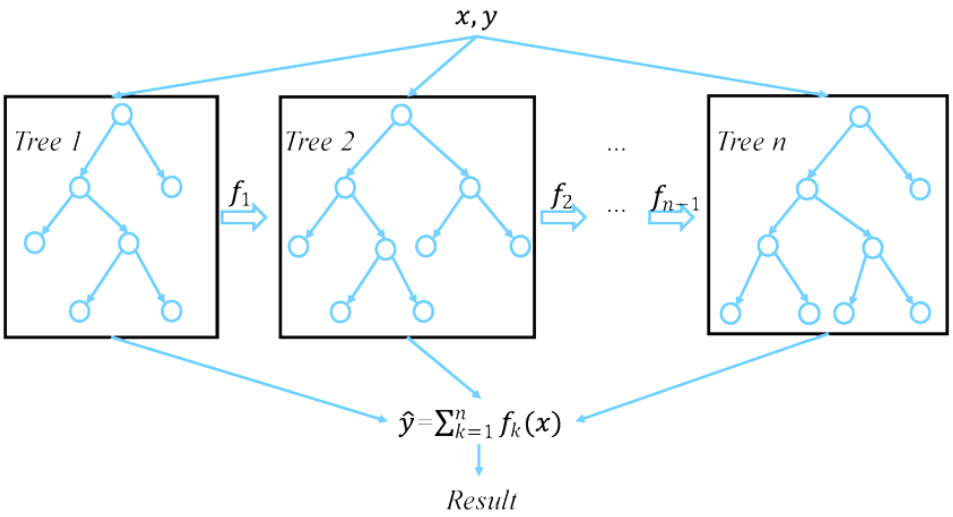


Figure A.7

A schematic diagram of the SVR [80]

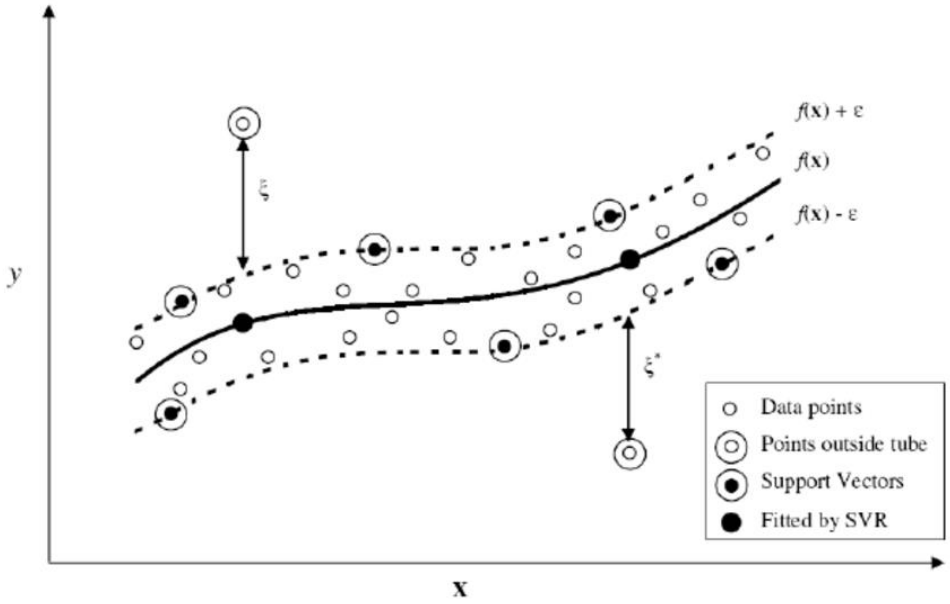


Figure A.8

Convolutional neural network (CNN) [81]

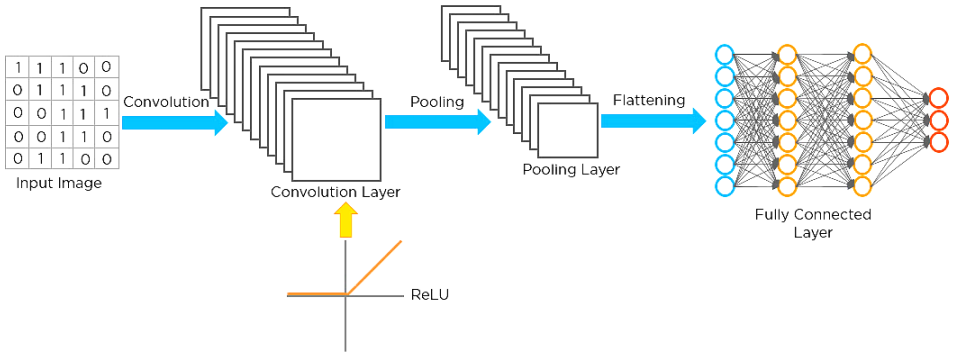


Figure A.9

Feature pyramid network (FPN)

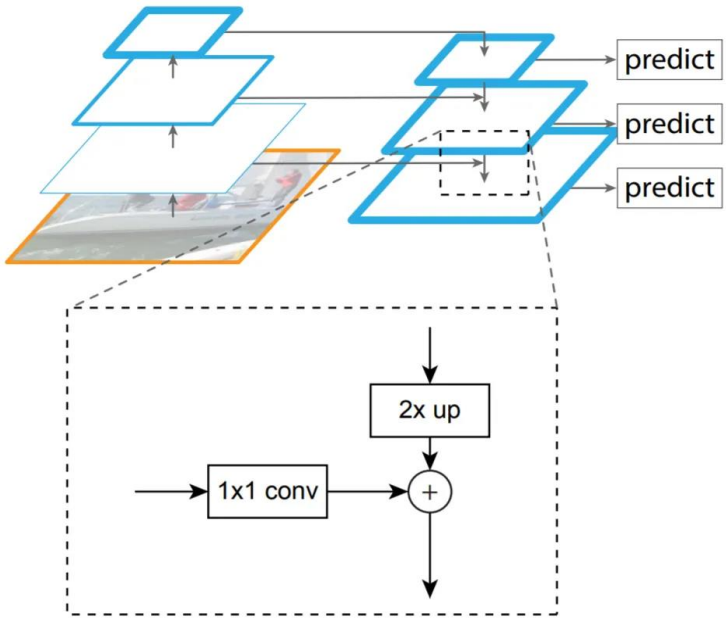


Figure A.10

Mask R-CNN architecture [82]

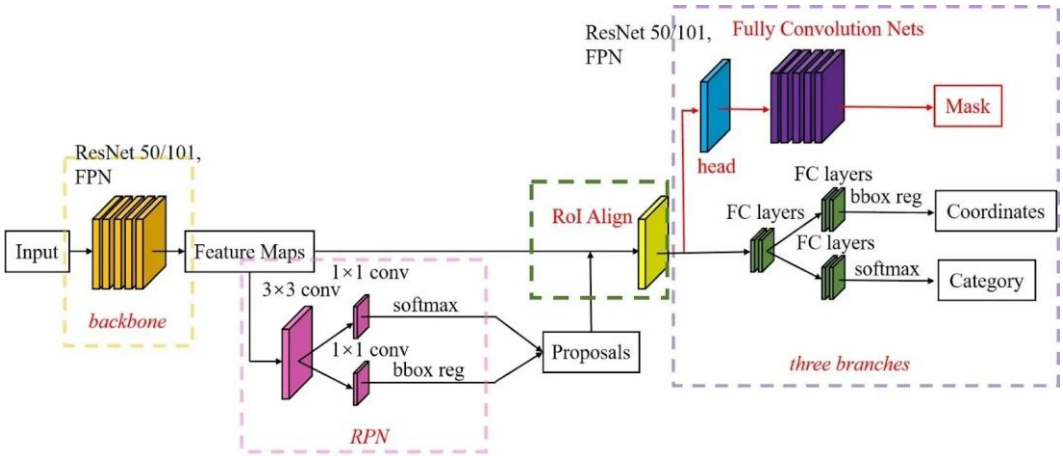


Figure A.11

YOLOv8 architecture [61]

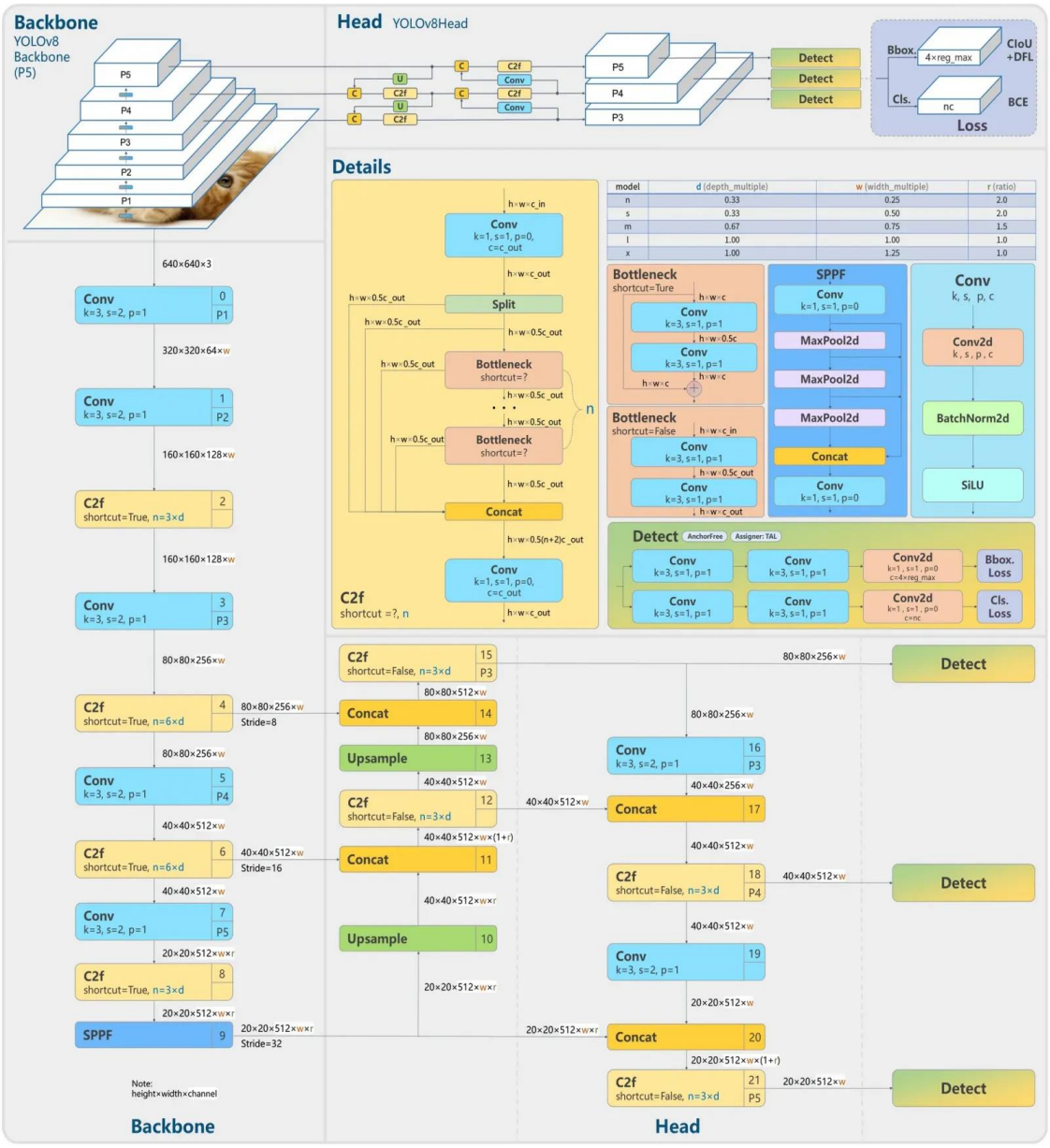


Figure A.12

Examples for 6 training samples

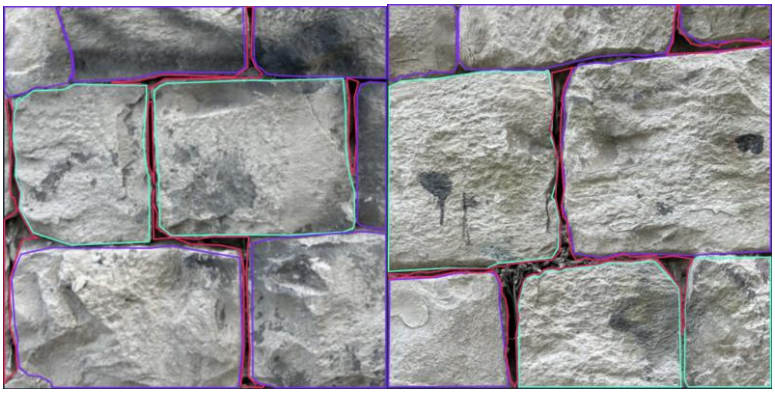
(a) (b)

(c) (d)

(e) (f)

Figure A.13

Ground truth annotations on 6 training samples



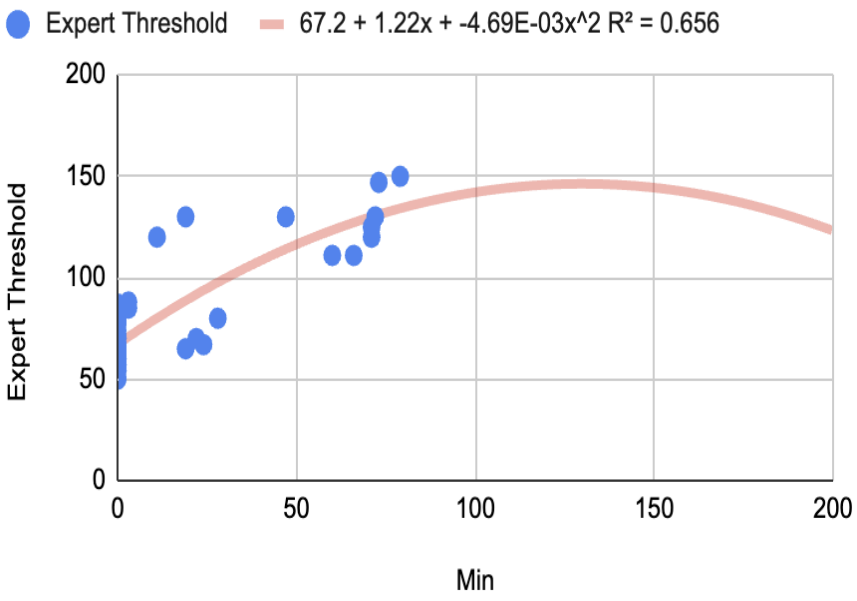
(a) (b)

(c) (d)

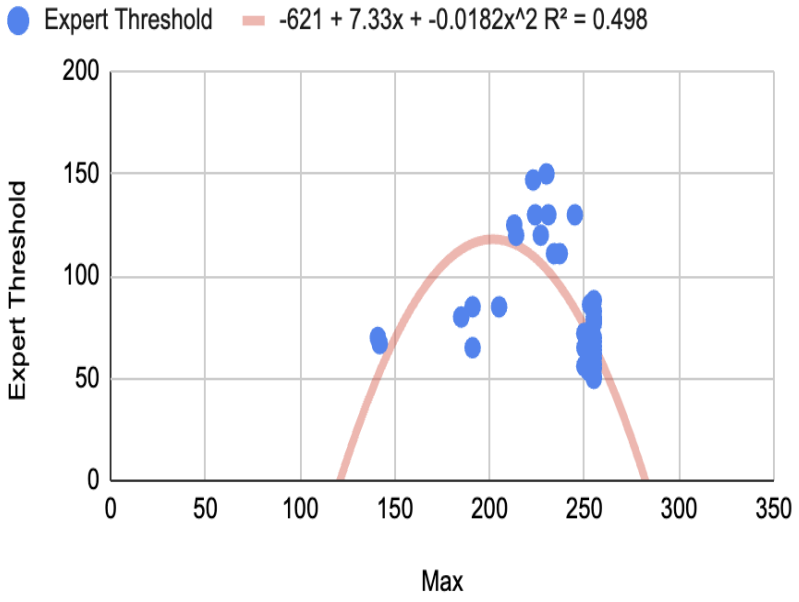
(e) (f)

Figure A.14

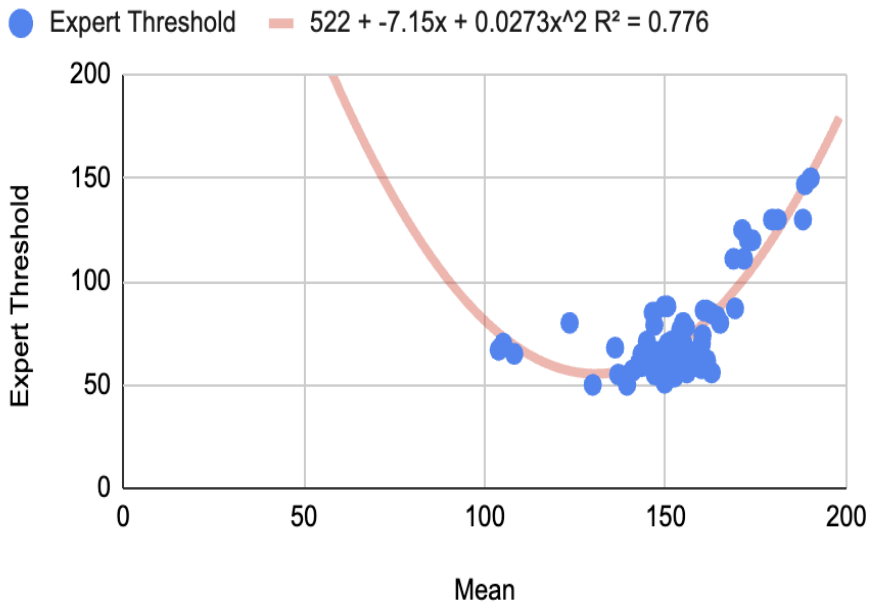
Comparison of polynomial regression models for seven histogram features



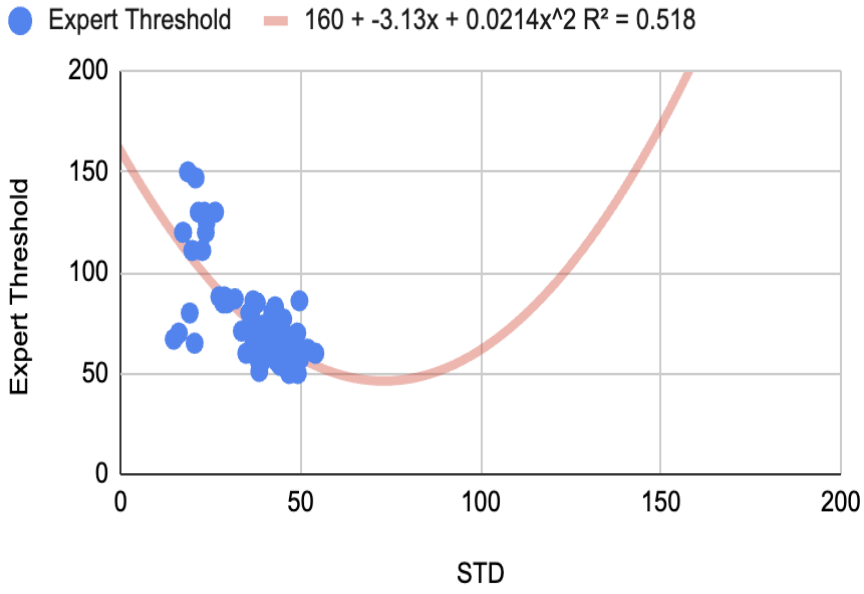
(a)



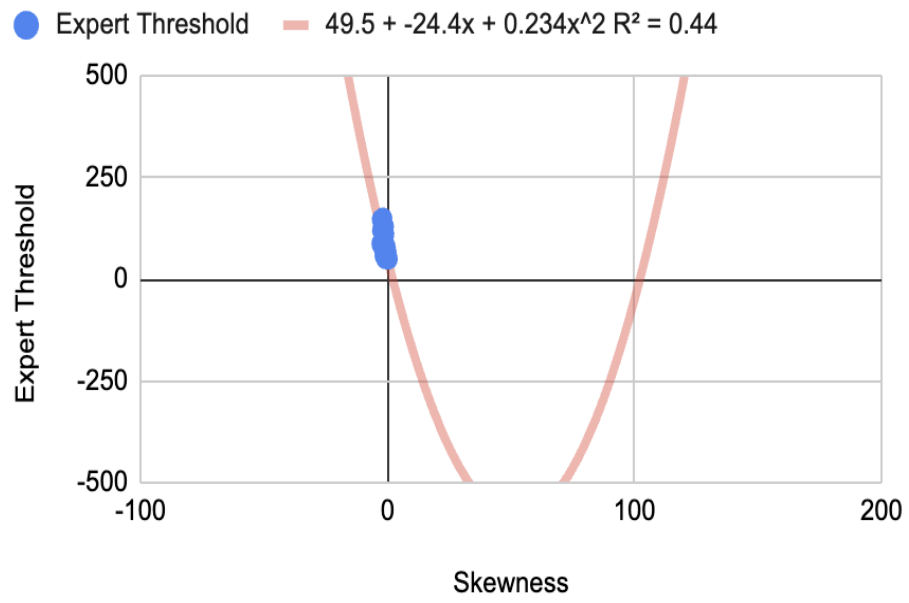
(b)


Expert Threshold
522 + -7.15x + 0.0273x^2 R² = 0.776
Expert Threshold
200
150
100
50
0
0
50
100
150
200
Mean

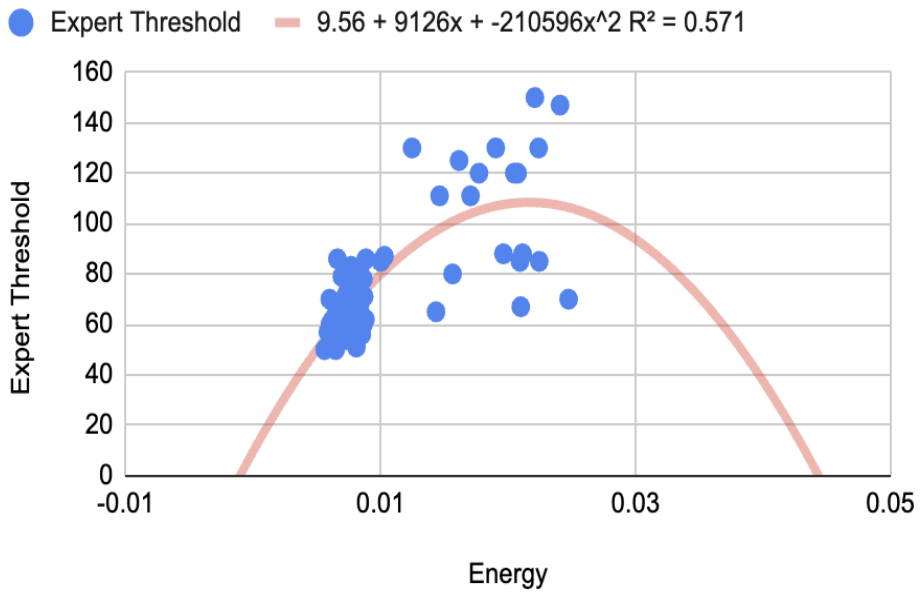
(c)


Expert Threshold
160 + -3.13x + 0.0214x^2 R² = 0.518
Expert Threshold
200
150
100
50
0
0
50
100
150
200
STD

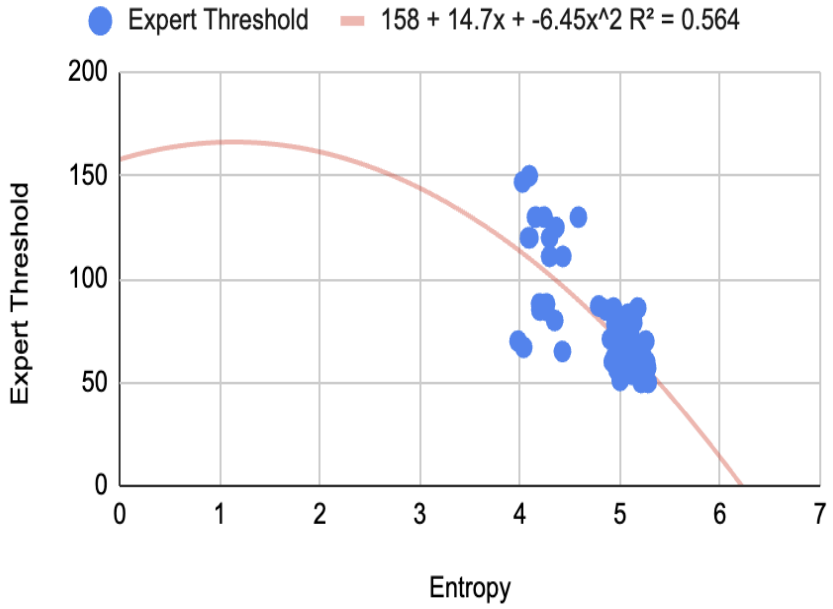
(d)


Expert Threshold
49.5 + -24.4x + 0.234x^2 R² = 0.44
500
250
0
-250
-500
-100
0
100
200
Expert Threshold
Skewness

(e)


Expert Threshold
9.56 + 9126x + -210596x^2 R² = 0.571
160
140
120
100
80
60
40
20
0
-0.01
0.01
0.03
0.05
Expert Threshold
Energy

(f)



(g)

Figure A.15

Distribution of the Threshold value in the dataset

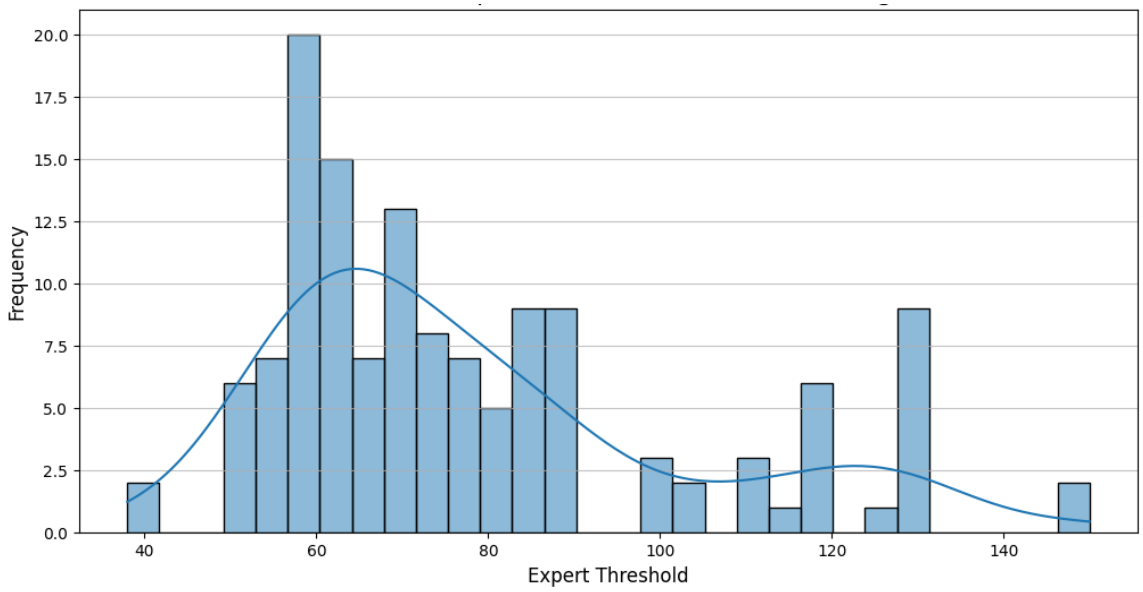


Figure A.16

Correlation matrix heatmap of all input features and Threshold value

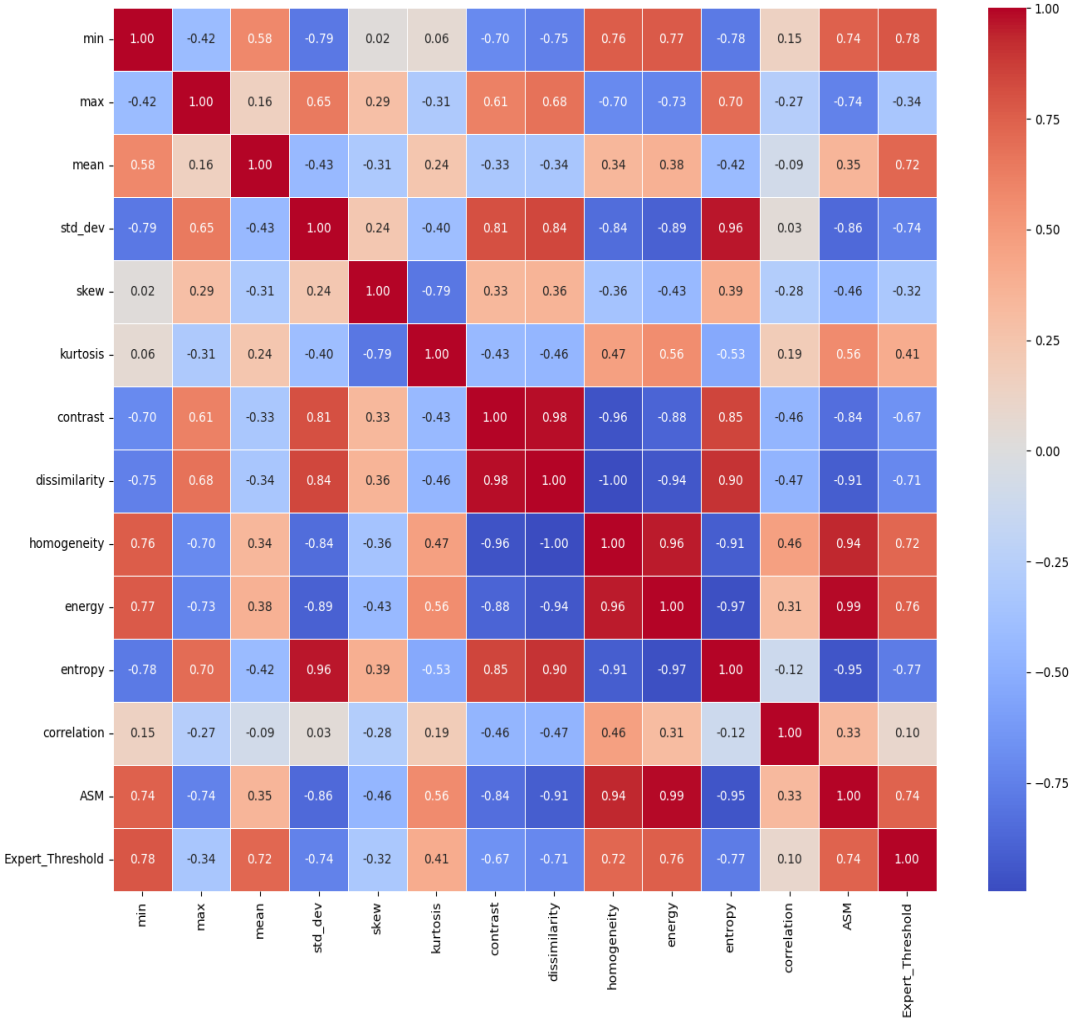
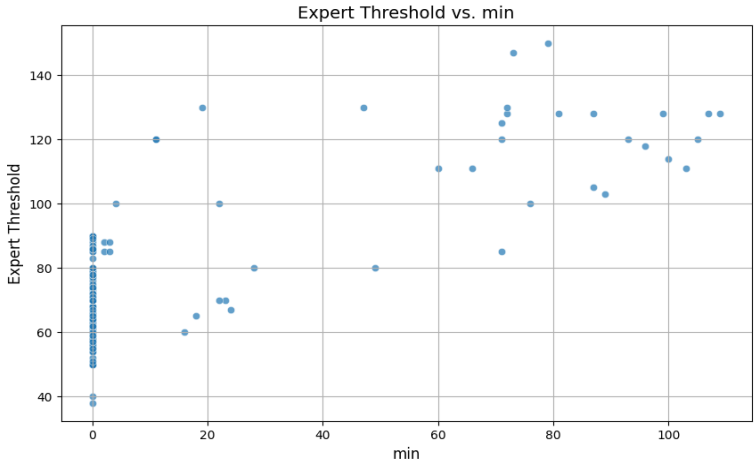
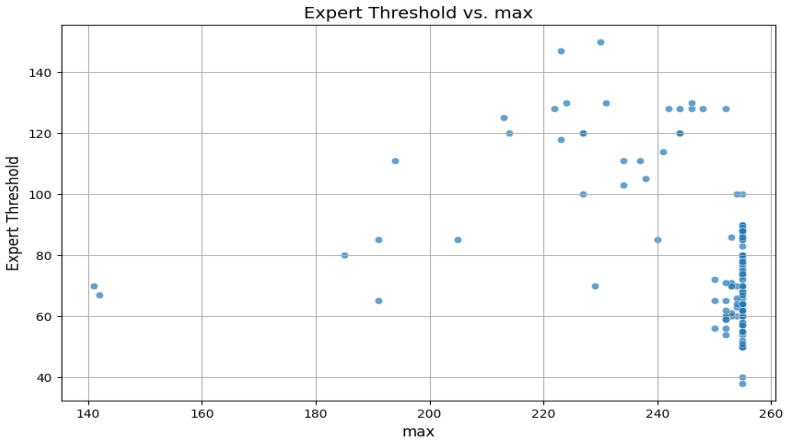


Figure A.17

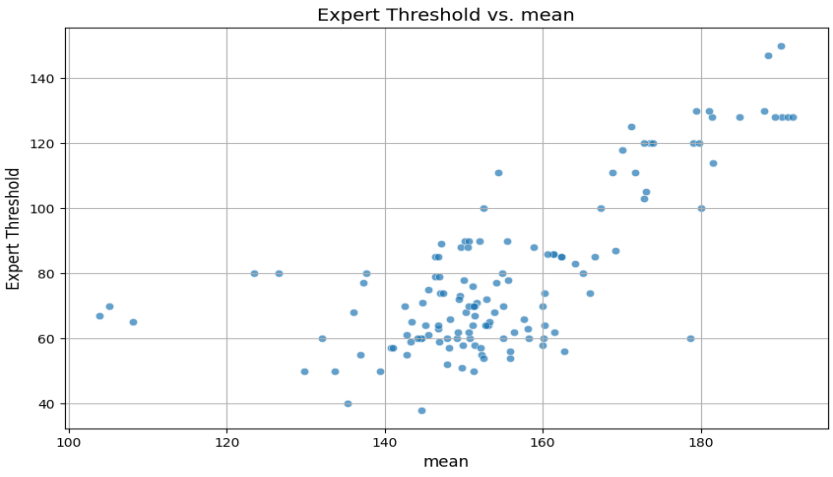
Analysis of feature-to-target relationships



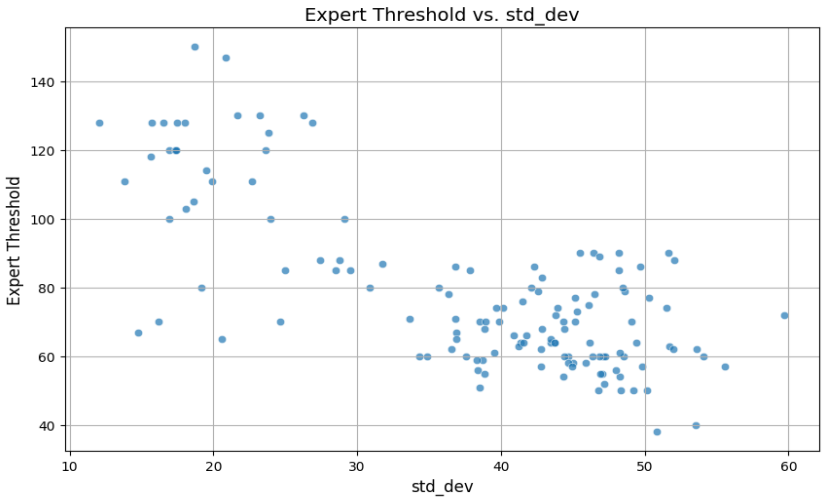
(a)



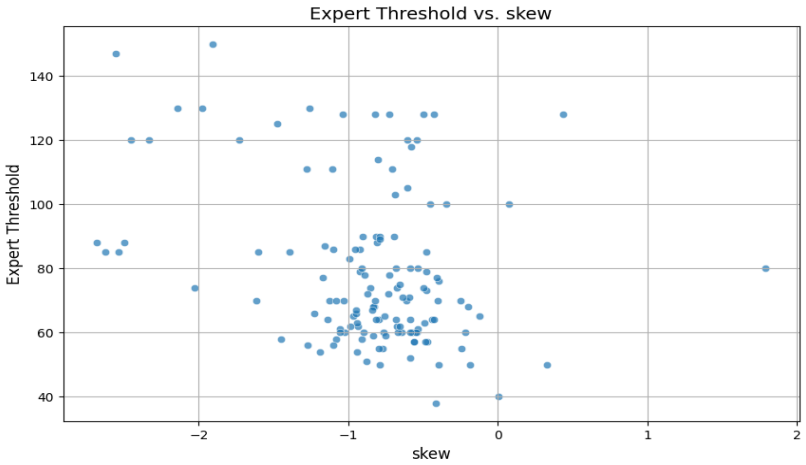
(b)



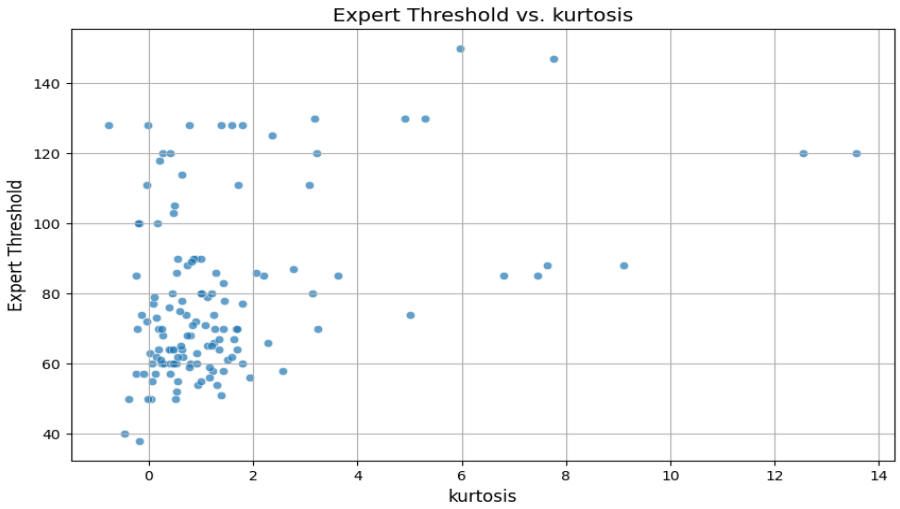
(c)


Expert Threshold vs. std_dev
Expert Threshold
std_dev
140
120
100
80
60
40
10
20
30
40
50
60

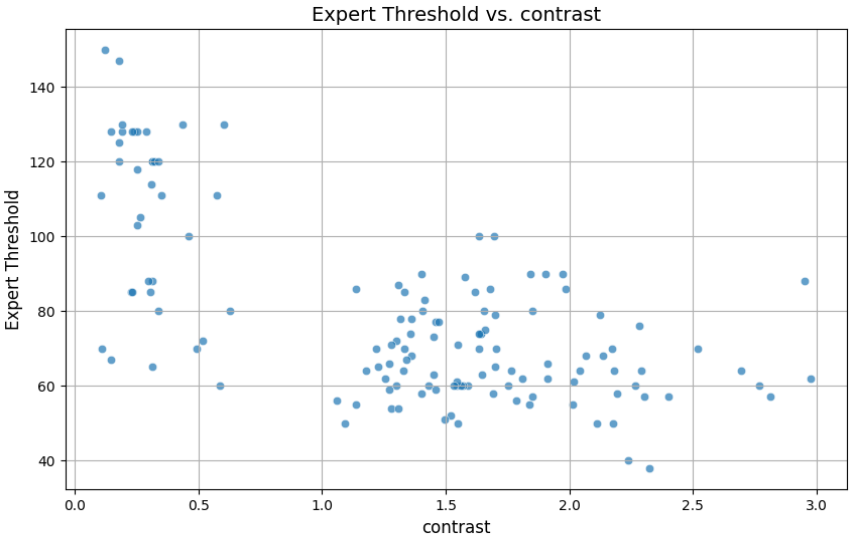
(d)


Expert Threshold vs. skew
Expert Threshold
skew
140
120
100
80
60
40
−2
−1
0
1
2

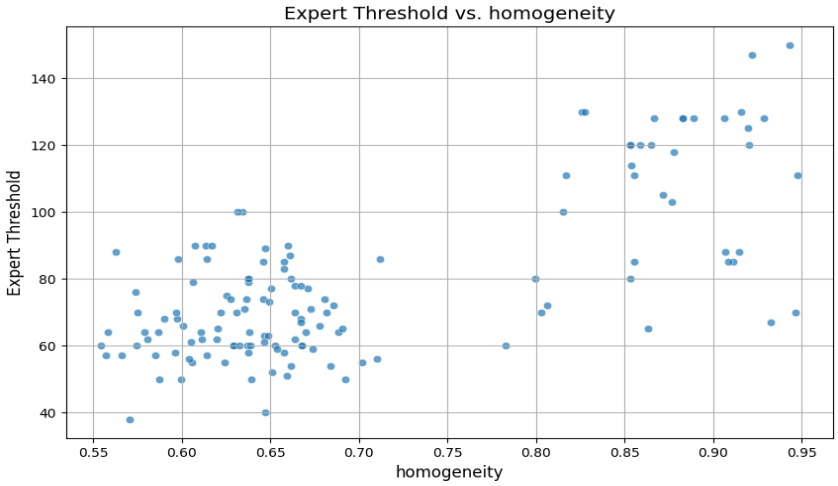
(e)


Expert Threshold vs. kurtosis
Expert Threshold
kurtosis
140
120
100
80
60
40
0
2
4
6
8
10
12
14

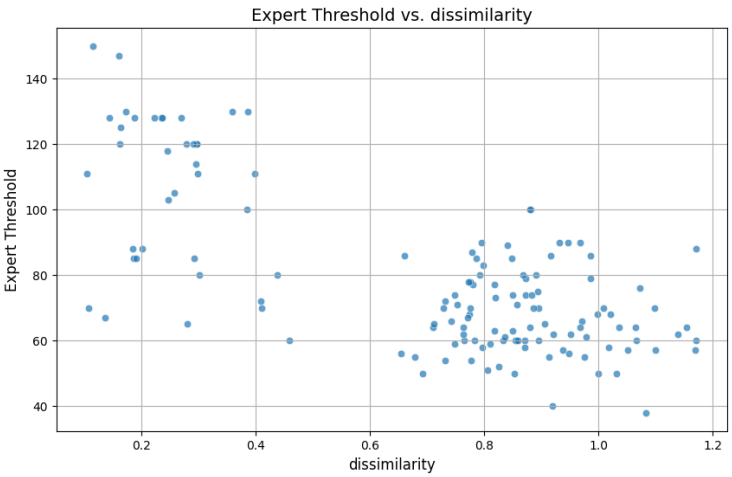
(f)


Expert Threshold vs. contrast
Expert Threshold
contrast
140
120
100
80
60
40
0.0
0.5
1.0
1.5
2.0
2.5
3.0

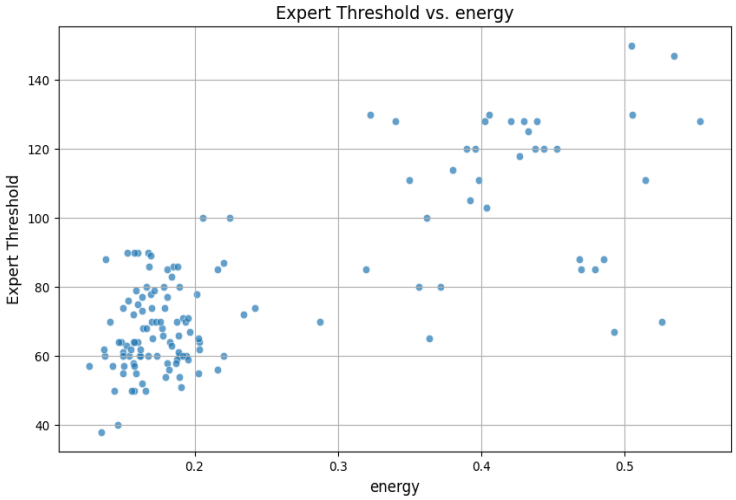
(g)


Expert Threshold vs. homogeneity
Expert Threshold
homogeneity
140
120
100
80
60
40
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95

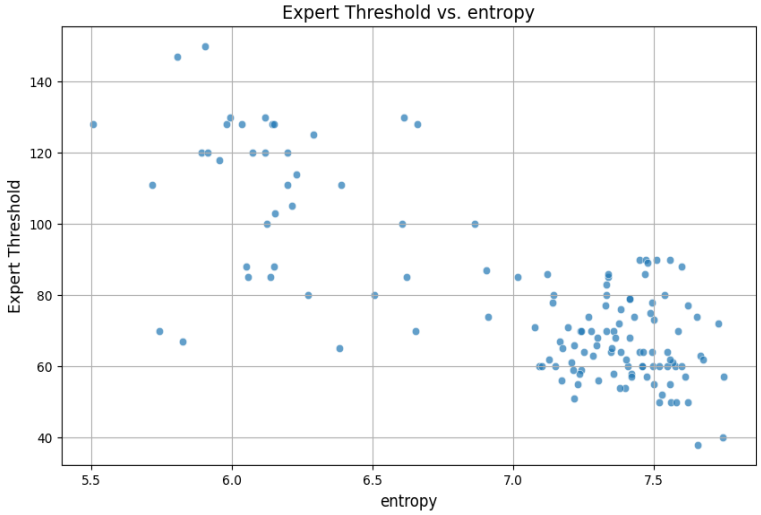
(h)


Expert Threshold vs. dissimilarity
Expert Threshold
dissimilarity
140
120
100
80
60
40
0.2
0.4
0.6
0.8
1.0
1.2

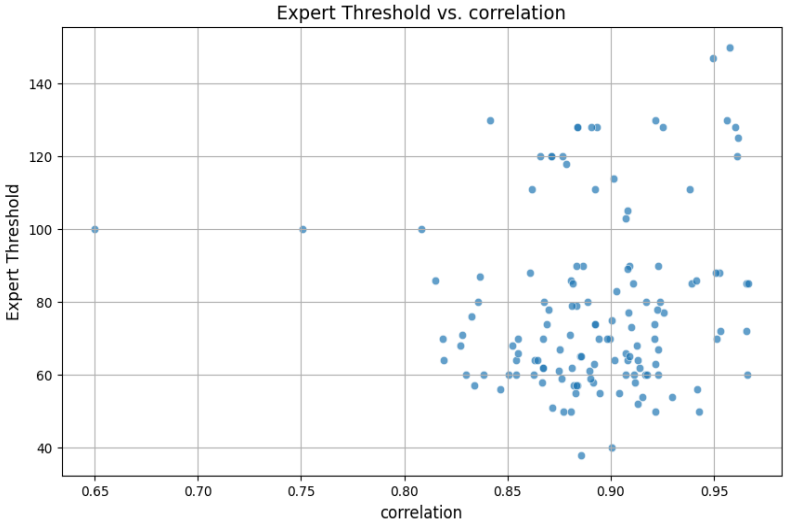
(i)


Expert Threshold vs. energy
Expert Threshold
energy
140
120
100
80
60
40
0.2
0.3
0.4
0.5

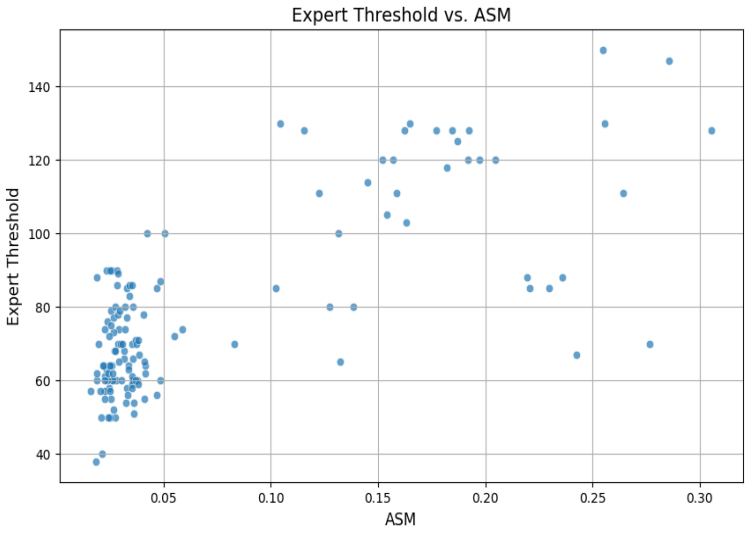
(j)


Expert Threshold vs. entropy
Expert Threshold
entropy
140
120
100
80
60
40
5.5
6.0
6.5
7.0
7.5

(k)


Expert Threshold vs. correlation
Expert Threshold
correlation
140
120
100
80
60
40
0.65
0.70
0.75
0.80
0.85
0.90
0.95

(l)



(m)

Figure A.18

ResNet-50 model architecture [83]

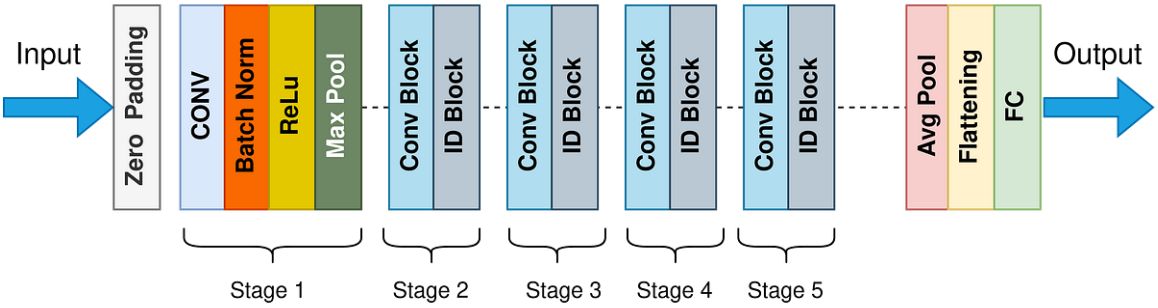


Figure A.19

Structure of backbone network of YOLOv8 [84]

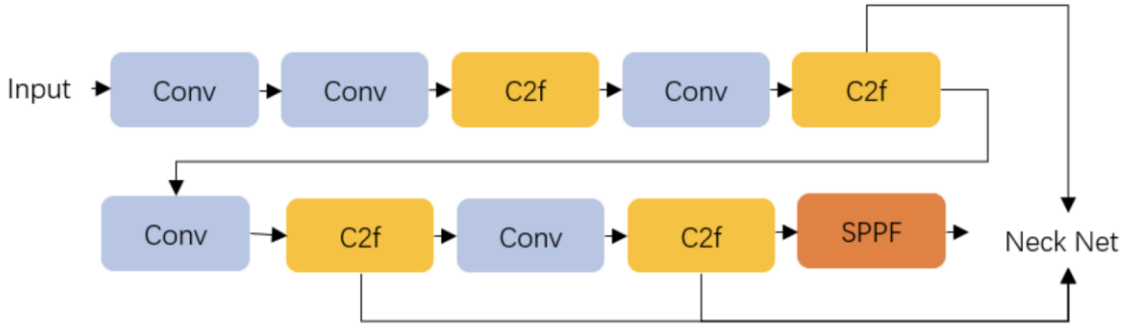


Figure A.20

Actual vs. Predicted Thresholds (Tuned XGBoost)

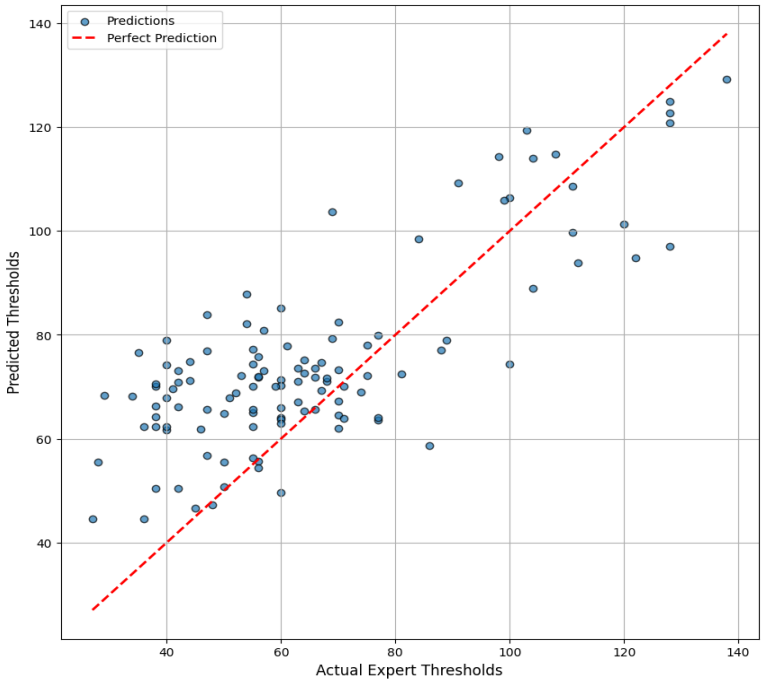


Figure A.21

Examples for 9 test samples

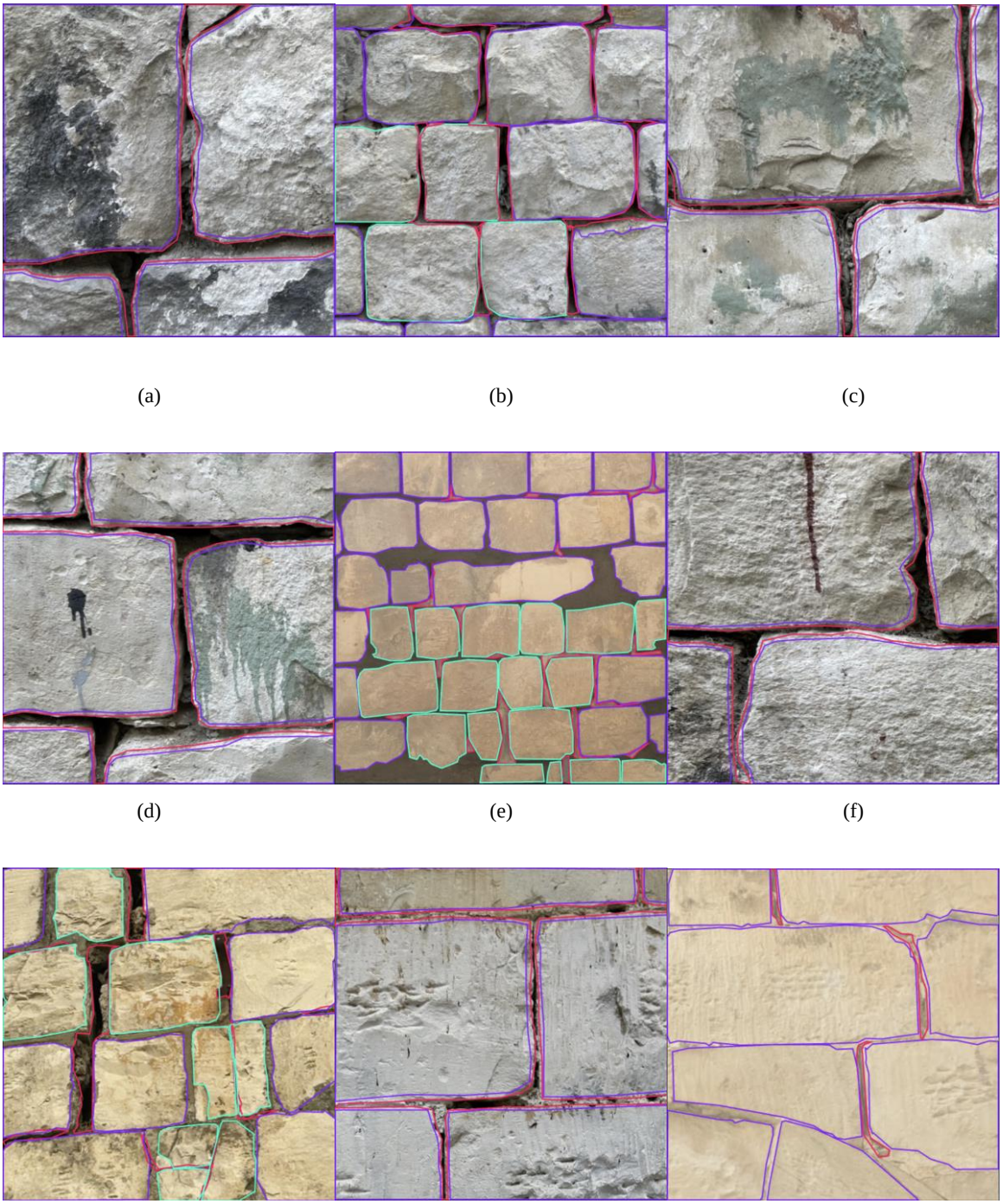
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A.22

Ground truth annotations on 9 test samples



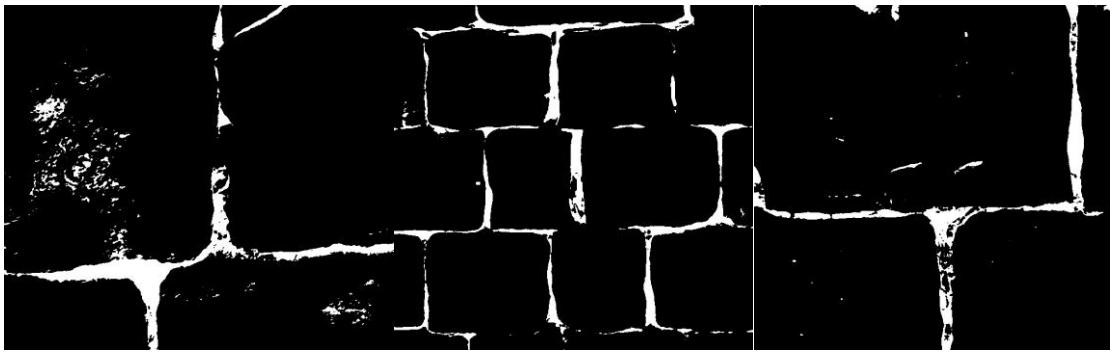
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A.23

Thresholding results by SDT on 9 test samples



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure A.24

Closing after SDT thresholding results on 9 test samples

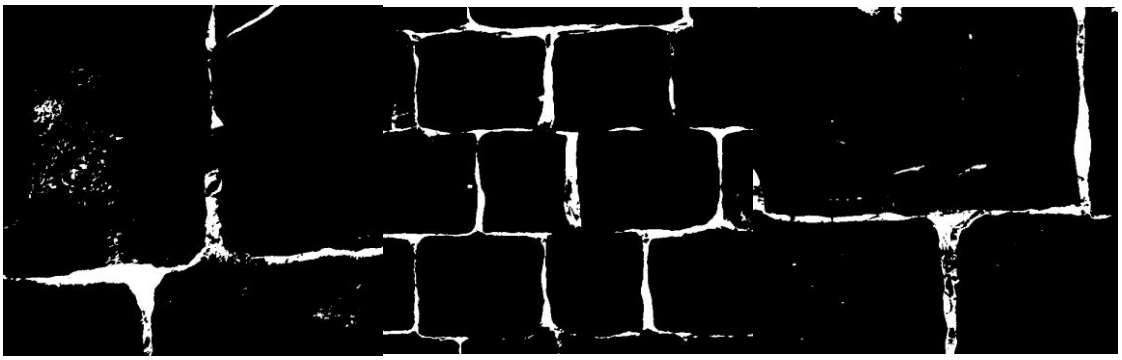
(a) (b) (c)

(d) (e) (f)

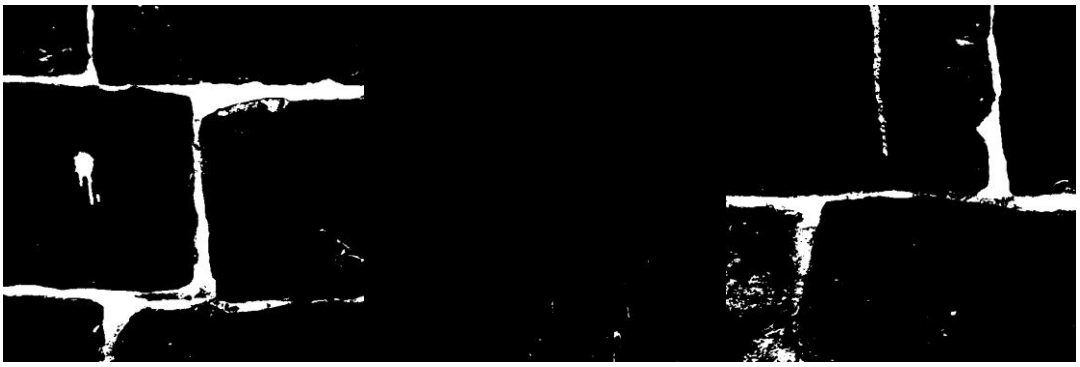
(g) (h) (i)

Figure A.25

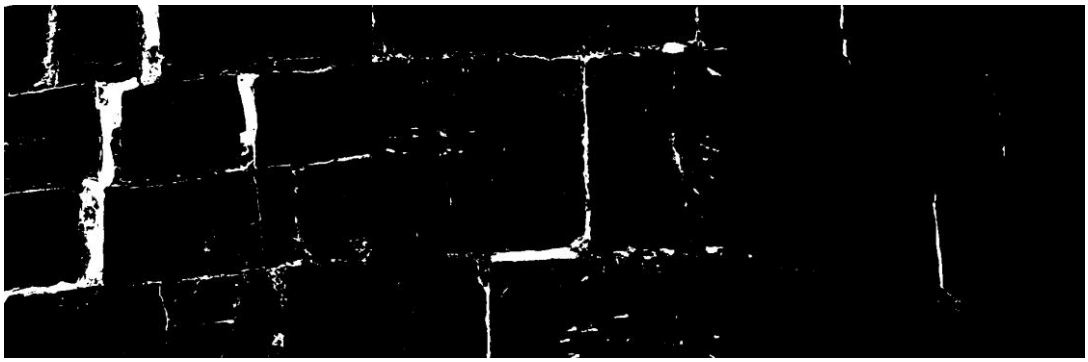
Thresholding results by ML on 9 test samples



(a) (b) (c)



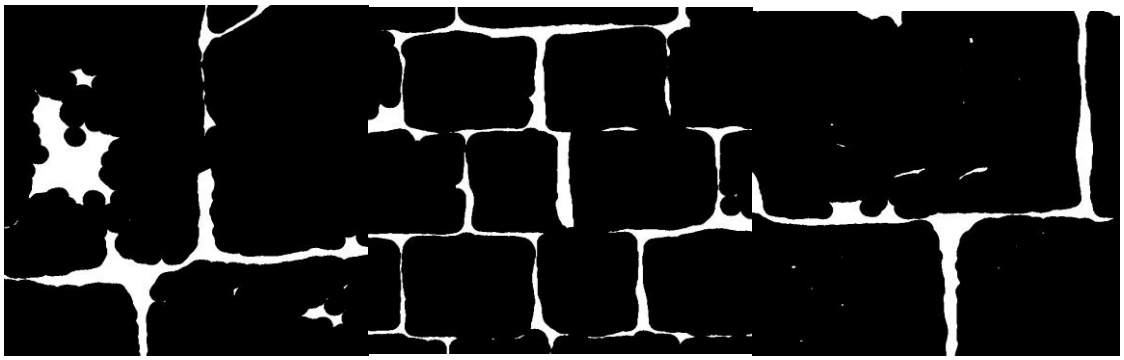
(d) (e) (f)



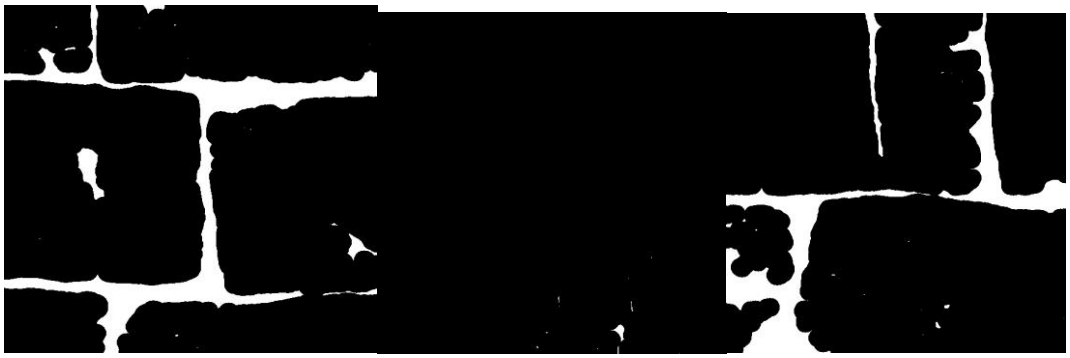
(g) (h) (i)

Figure A.26

Closing after ML thresholding results on 9 test samples



(a) (b) (c)

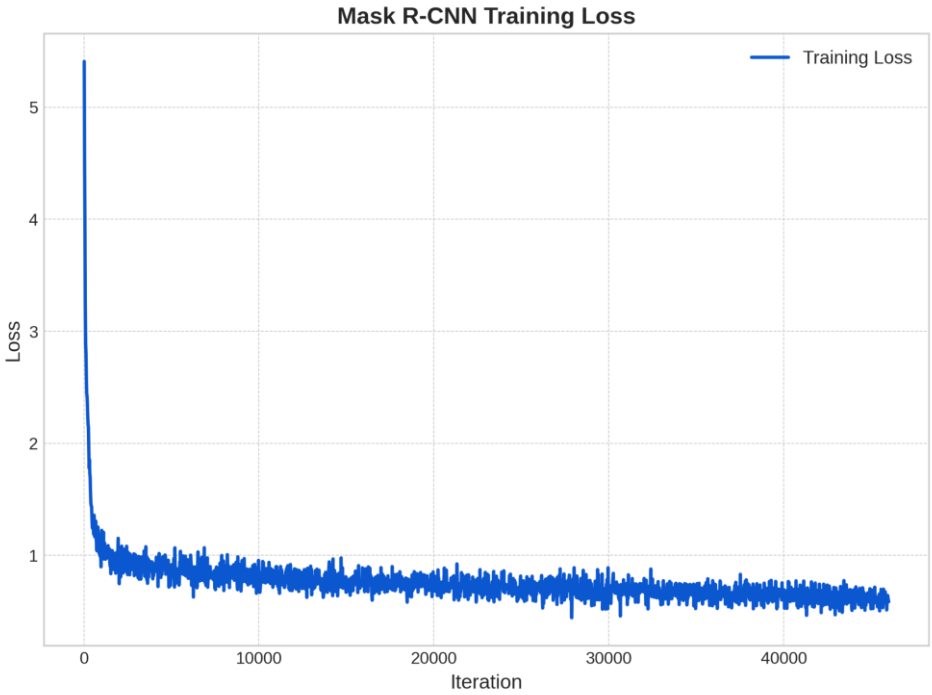


(d) (e) (f)

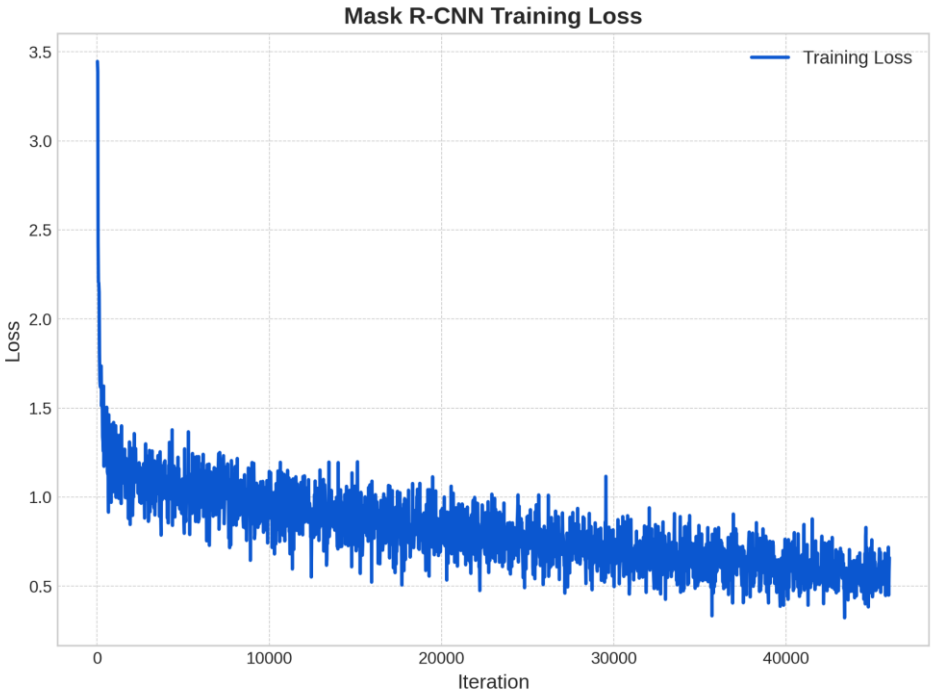
(g) (h) (i)

Figure A.27

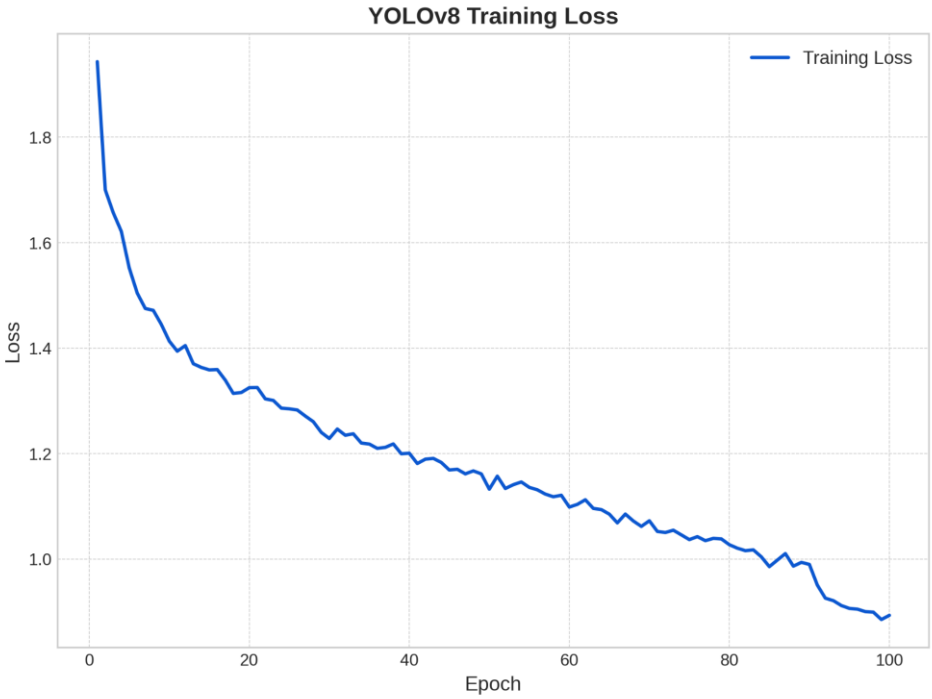
Training loss for single-class and multi-class training



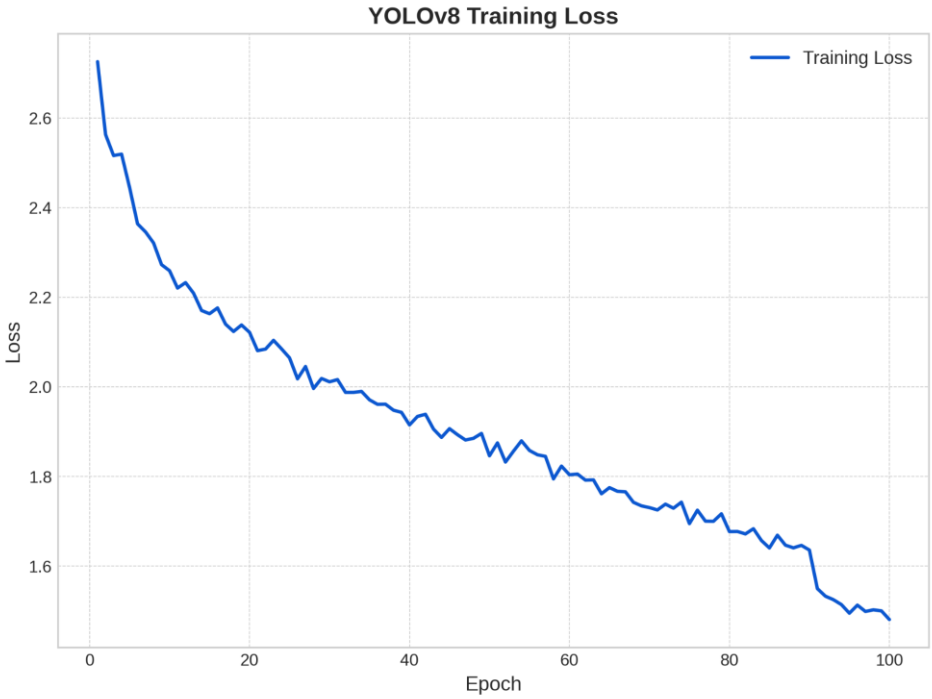
(a) Single-class



(b) Multi-class



(c) Single-class



(d) Multi-class

Figure A.28

Impact of single-class vs. multi-class training

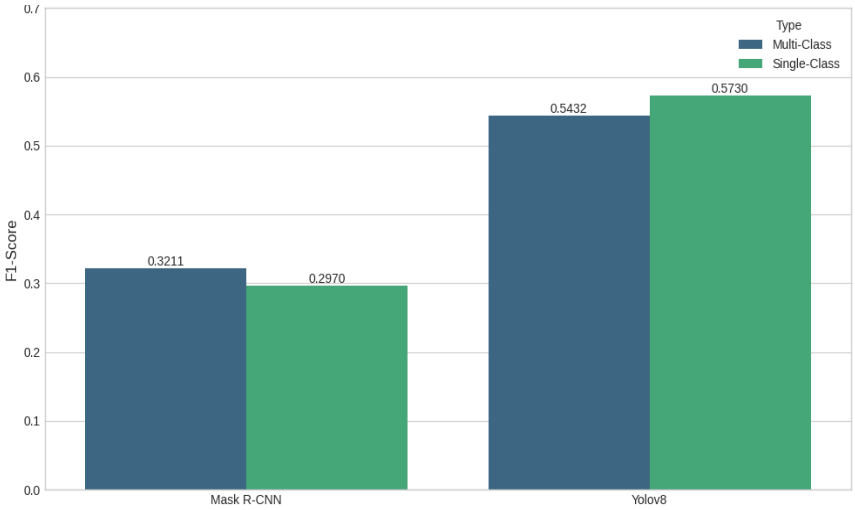


Figure A.29

Overall performance ranking by F1-Score

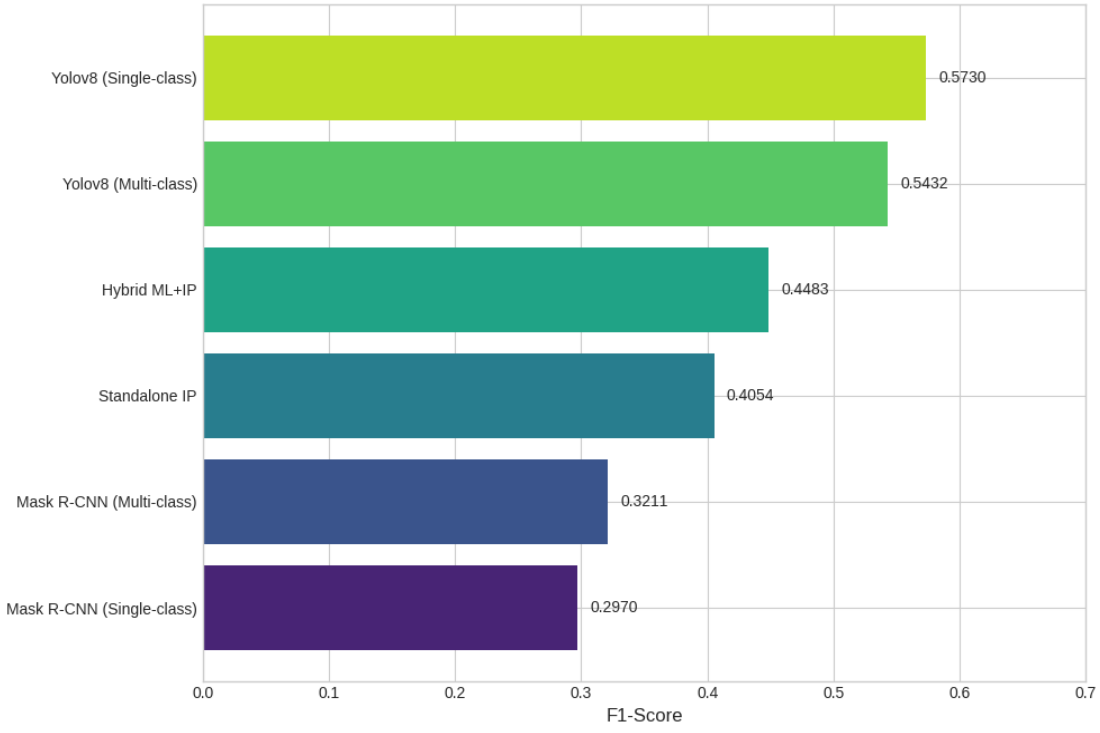


Figure A.30

Precision vs Recall trade-off

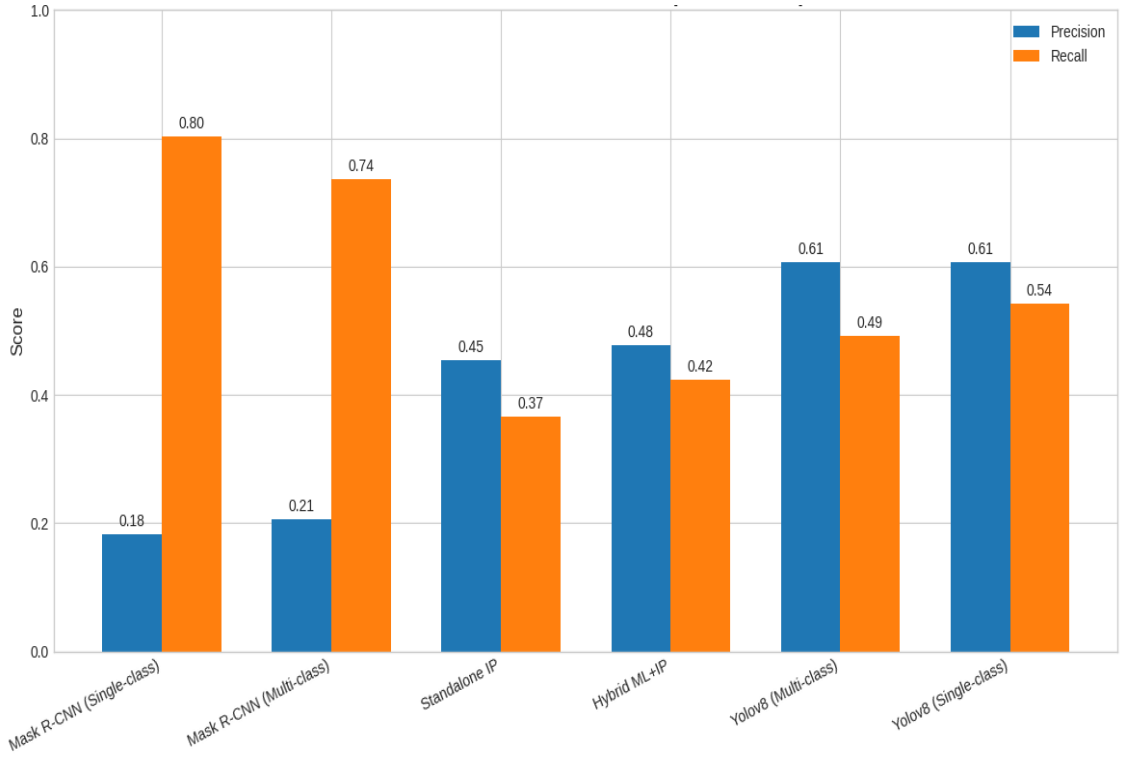
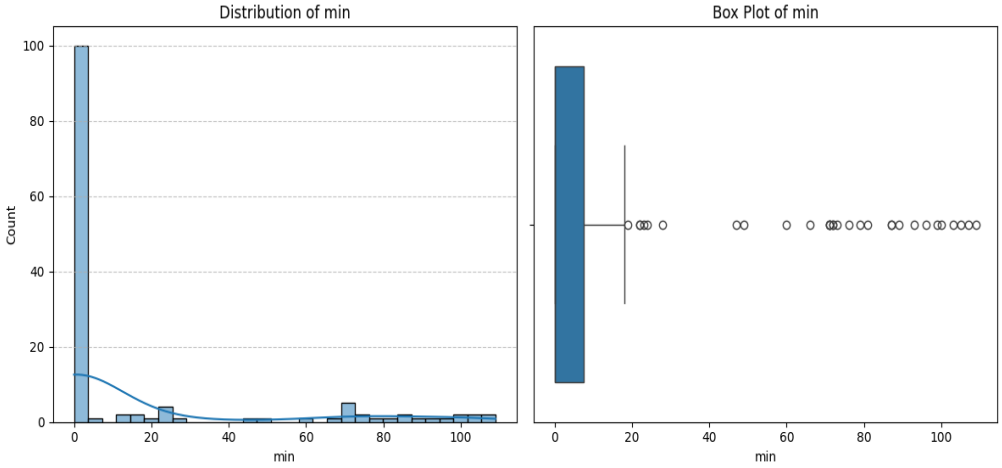
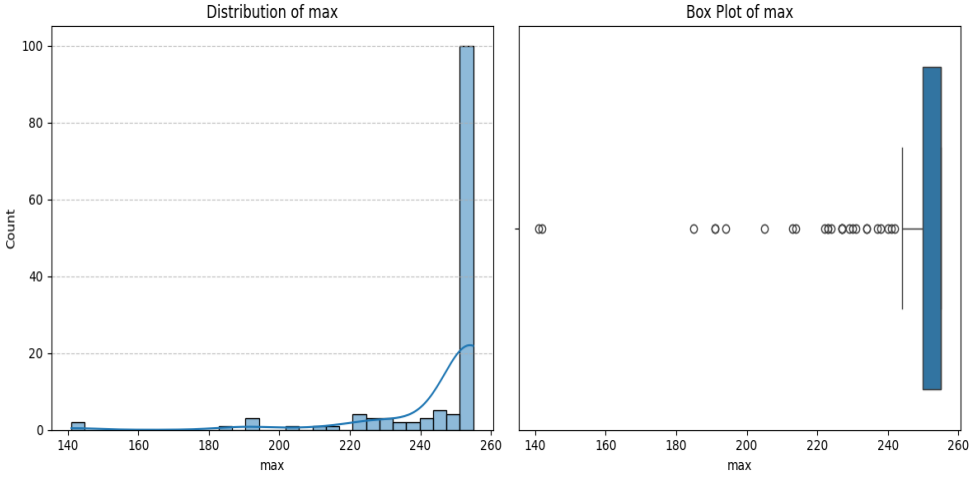


Figure A.31

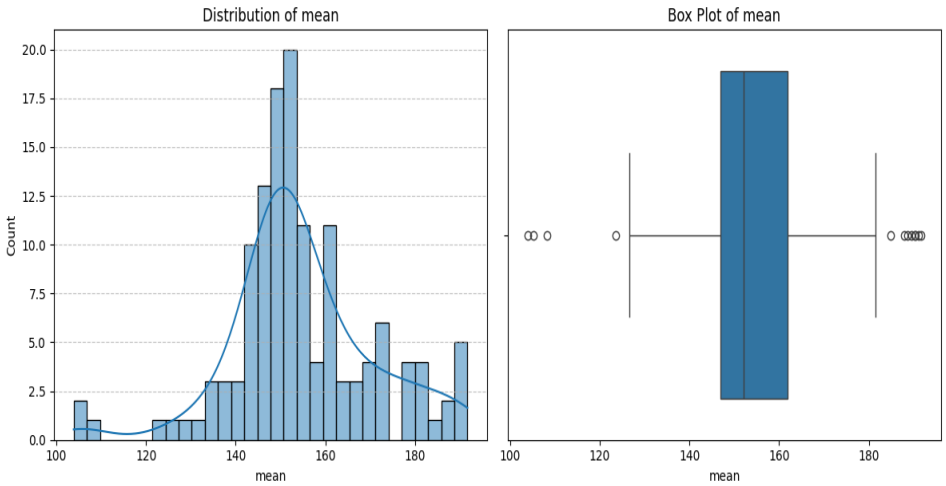
Distributions of seven histogram features



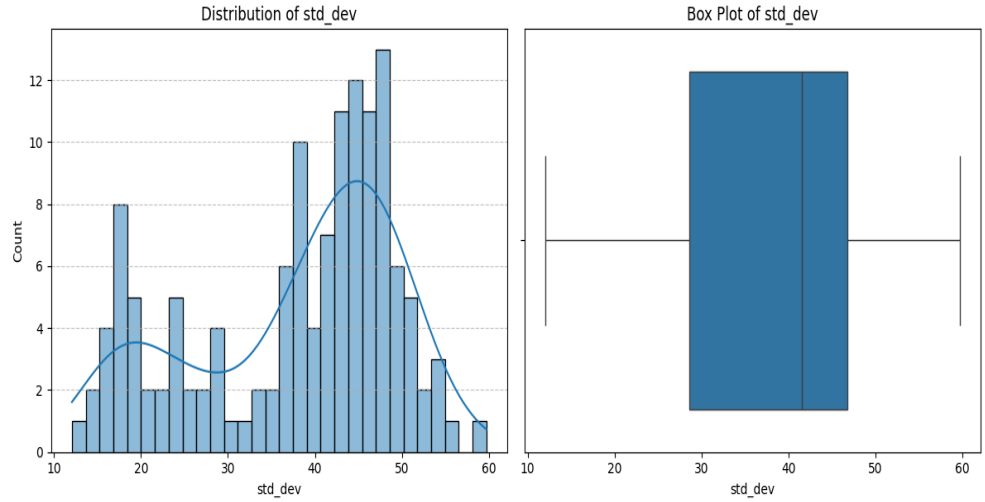
(a)



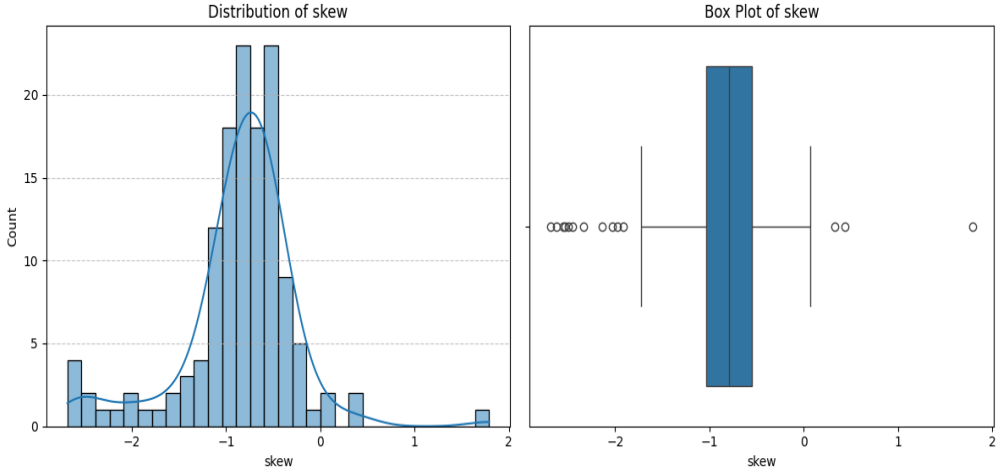
(b)



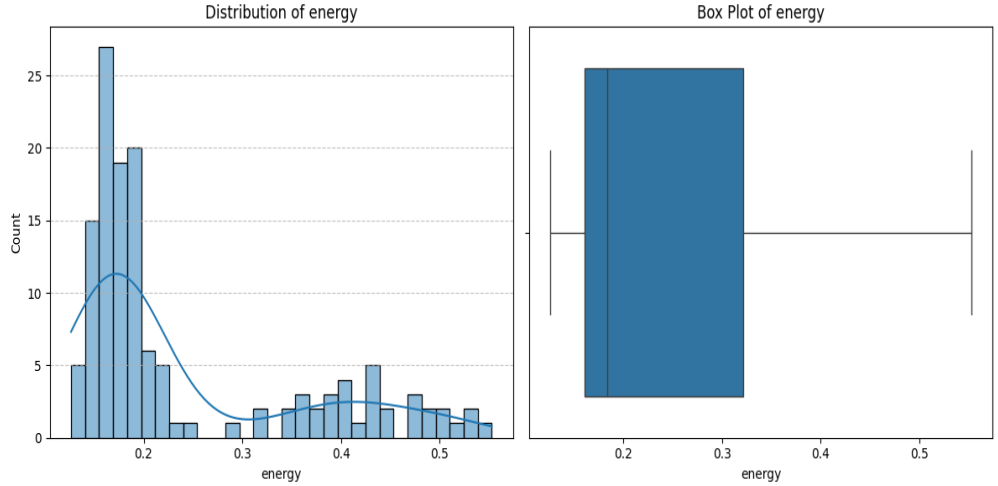
(c)


Distribution of std_dev
Box Plot of std_dev
Count
std_dev
std_dev

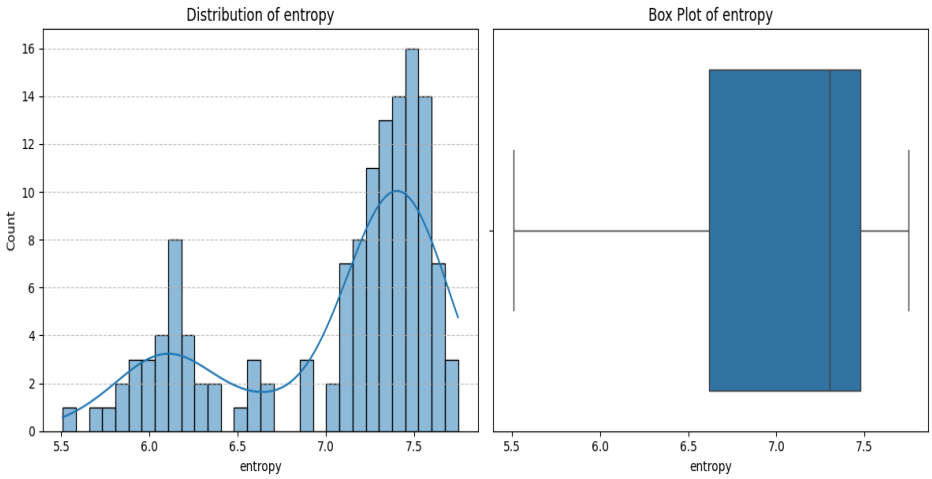
(d)


Distribution of skew
Box Plot of skew
Count
skew
skew

(e)


Distribution of energy
Box Plot of energy
Count
energy
energy

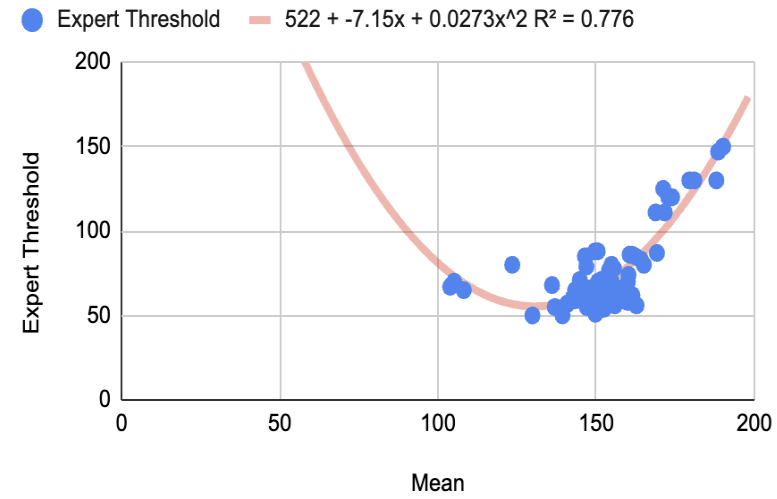
(f)



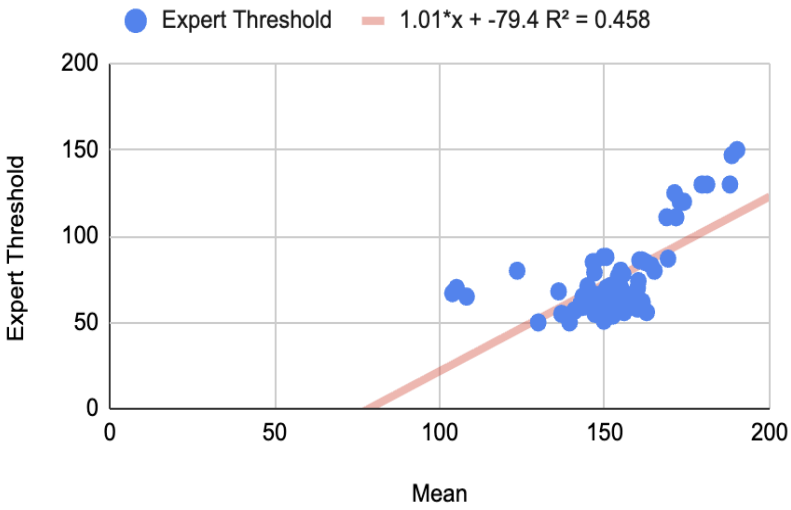
(g)

Figure A.32

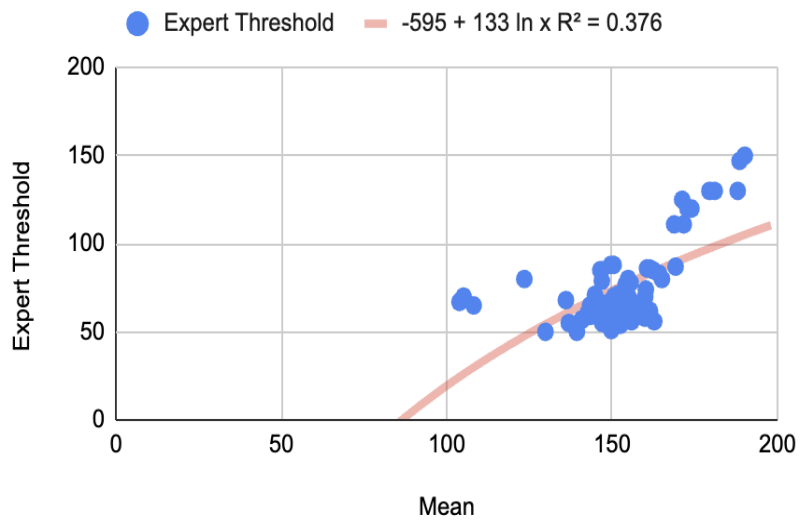
Comparison of regression models for the mean.



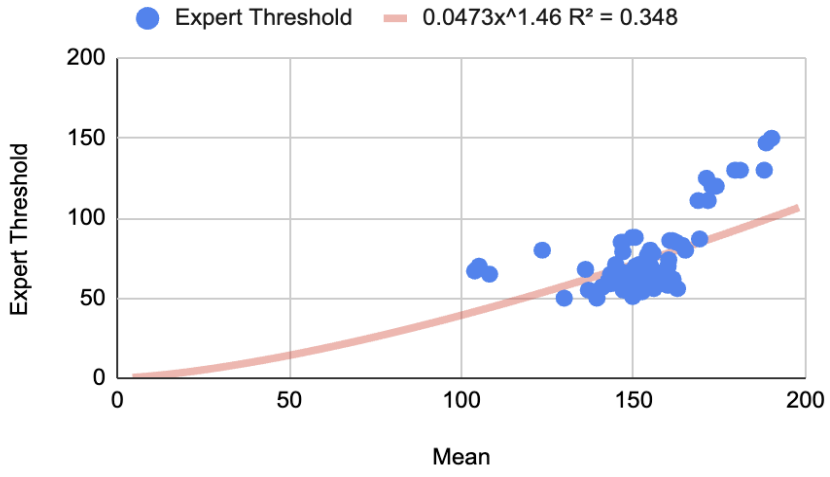
(a - Polynomial)



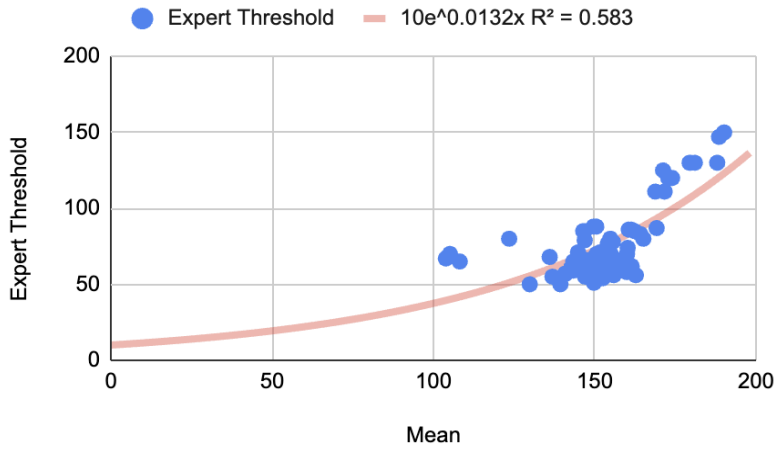
(b - Linear)



(c - Logarithmic)



(d - Power series)



(e - exponential)

Appendix B

Tables

Table B.1

The kernel of the Gaussian filter

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

Table B.2

Comparative analysis of proposed dataset (NADER-Crack) with publicly available benchmark datasets

Dataset name	Material	Annotation type	Classes	Number of images
SDNET2018 [85]	Concrete	Classification	Binary Crack, No Crack	56,000
CFD (Crack Forest) [86]	Pavement (Asphalt)	Semantic segmentation	1 Crack	118
Crack500 [87]	Pavement (concrete)	Semantic segmentation	1 Crack	3,368
DeepCrack [88]	Multi-surface	Semantic segmentation	1 Crack	537
Masonry-Crack-2019 [89]	Masonry (brick)	Classification	1 Crack	5000
NADER-Crack	Masonry (brick)	Instance segmentation	3 Crack, broken_brick, brick	2,794

Table B.3

Comparative R^2 analysis of regression models for the Mean

	Linear	Logarithmic	Power series	Exponential	Polynomial
R^2 (Mean)	0.45	0.37	0.34	0.58	0.77

جامعة النجاح الوطنية

كلية الدراسات العليا

الكشف عن التشققات والتقييم في المباني التاريخية باستخدام الذكاء الاصطناعي

إعداد

لين ابراهيم ذياب أبوشقير

إشراف

د. أنس طعمة

د. محمد سماعنة

قدمت هذه الرسالة استكمالا لمتطلبات الحصول على درجة الماجستير في الذكاء الاصطناعي، من كلية الدراسات العليا، في جامعة النجاح الوطنية، نابلس– فلسطين.

2026

الكشف عن التشققات والتقييم في المباني التاريخية باستخدام الذكاء الاصطناعي

إعداد

لين ابراهيم ذياب أبوشقير

إشراف

د. أنس طعمة

د. محمد سماعنة

الملخص

برزت تقنيات الذكاء الاصطناعي كأدوات فعالة لأتمتة كشف الشقوق في المباني التاريخية. وفي سياق ضمان السلامة الهيكلية للمباني الحجرية التاريخية، تقدم هذه التقنيات حلاً فعالاً.تعد هذه المهمة بالغة الأهمية للسلامة والصيانة، حيث يمكن للكشف الاستباقي عن الشقوق أن يمنع الأضرار و المخاطر. ومع ذلك، لا يزال الكشف عن الشقوق في هذه الهياكل يشكل تحدياً كبيراً.

تستكشف هذه الرسالة أساليب الكشف الآلي الفعالة عن الشقوق باستخدام الذكاء الاصطناعي، حيث تدرس ثلاثة مناهج مختلفة للرؤية الحاسوبية. يركز النهج الأول على نموذج إحصائي يعتمد على صيغ رياضية ثابتة وخصائص نسيجية لفصل الشقوق عن الخلفية. ثم تم تطوير هذا النهج باستخدام نموذج محسن قائم على التعلم الآلي، حيث تستخرج الخصائص الإحصائية والنسيجية للتنبؤ. وأخيرا، يعتمد النهج الثالث على استخدام نماذج التعلم العميق باستخدام التعلم الانتقالي.

لدعم متطلبات نموذج التعلم العميق كثيفة البيانات، تضمن البحث جمع بيانات مكثف في البلدة القديمة بنابلس. تم جمع البيانات من مواقع متعددة في جميع أنحاء المدينة لضمان تنوع الظروف الإنشائية وأنماط التشققات ونتيجةً لذلك، تم إنشاء مجموعة بيانات تضم 2794 صورة لهياكل حجرية. وقُيّمت جميع الطرق باستخدام مجموعة اختبار متسقة لضمان مقارنات نزيهة.

صُممت منهجية التقييم بالتعاون مع خبراء في الهندسة المدنية لتقييم الأداء عبر بُعدين رئيسيين: F1-Score ومعدل الكشف. يعكس F1-Score موثوقية النماذج، ويعكس معدل الكشف ضمان السلامة. أظهرت النتائج تسلسلاً هرمياً واضحاً في الأداء. حقق نموذج الكشف عن الشقوق القائم على الإحصاءات درجة F1-Score أساسية قدرها 0.4775. وحسّن النموذج القائم على التعلم الآلي ذلك بدرجة F1-Score قدرها 0.5151، مؤكداً بذلك ميزة تحسين المعلمات القائم على البيانات. في مجال التعلم العميق، أظهر نموذج YOLOv8 موثوقية جيدة، محققاً أعلى درجة F1-Score وهي 0.6116، متوازناً بشكل فعال بين الدقة والحساسية. ومع ذلك، عند تقييمه على معدل الكشف، أثبت Mask R-CNN أداءً متفوقاً، حيث حدد 89.23٪ من جميع الشقوق المحتملة.

الكلمات المفتاحية: كشف الشقوق، المنشآت التاريخية، معالجة الصور، تعلم الآلة، التعلم العميق، التعلم بالنقل.