# PAVEMENT CRACK DETECTION AND CHARACTERIZATION USING DEEP LEARNING AND PIXEL LEVEL SEGMENTATION

BY

ARSELAN ASHRAF

A thesis submitted in fulfillment of the requirement for the degree of Doctor of Philosophy in Engineering

Kulliyyah of Engineering
International Islamic University Malaysia

September 2024

# ABSTRACT

Pavement cracks pose a significant threat to road safety and infrastructure integrity, leading to potential hazards for vehicles and necessitating costly repairs. The traditional methods for detecting and characterizing these cracks are often manual, time-consuming, and subject to human error, making it challenging to maintain roads efficiently and effectively. This study advances pavement maintenance by applying deep learning and pixel-level segmentation to improve the detection and characterization (classification and sizing) of pavement cracks, crucial for road infrastructure safety and longevity. The research began with extensive data collection, using a GoPro Hero 8 mounted on a vehicle to gather images of roads in Kuala Lumpur and Selangor, Malaysia. Following detailed preprocessing, including image augmentation, blurring, and light intensity variations, a pavement crack dataset was prepared for analysis. The investigation started with a customized YOLOv7 model, achieving 0.9545 recall and 0.9523 precision on the RDD2022 dataset and 0.9158 recall and 0.93 precision on our custom dataset. When benchmarked against traditional methods, such as ConvNets and deep neural networks, the customized YOLOv7 model demonstrated better precision and recall performance. Subsequent work with the YOLOv8x-seg model resulted in improved precision and recall in crack detection, with performance metrics of 0.93 recall and 0.91 precision for alligator cracks, 0.95 recall and 0.84 precision for longitudinal cracks, and 0.806 recall and 0.89 precision for transverse cracks. In direct benchmarking, this model surpassed previous fusion-based deep convolutional methods in both precision and recall. The final phase involved creating an Advanced Hybrid Deep Learning Model incorporating Deep Gradient ResNet and a Modified Attention mechanism, enhancing crack detection and characterization. This model showed a promising precision of 0.84 for alligator cracks, 0.89 for longitudinal cracks, and 0.87 for transverse cracks, with recall rates of 0.96 for alligator cracks, 0.88 for longitudinal cracks, and 0.80 for transverse cracks. The developed model outperformed the benchmarked research utilizing CrackNet model in both precision and recall. Utilizing advanced segmentation and machine vision techniques, the study successfully demonstrated the capability to precisely size pavement cracks, which is vital for targeted maintenance. The research signifies progress in automated pavement crack analysis, contributing to safer and more durable road infrastructures.

# خلاصة البحث

تشكل شقوق الرصيف تهديداً كبيراً لسلامة الطرق وسلامة البنية التحتية، وهو ما يؤدي إلى مخاطر محتملة على المركبات، ويستلزم إصلاحات مكلفة. وغالباً ما تكون الطرق التقليدية لاكتشاف هذه الشقوق وتوصيفها يدوية، وتستغرق وقتاً طويلاً، وتكون عرضة للخطأ البشري، وتلك العوامل تجعل من الصعب صيانة الطرق بكفاءة وفعالية. تعمل هذه الدراسة على تحسين عملية صيانة الرصيف من خلال تطبيق التعلم العميق (Deep Learning) والتجزئة على مستوى البكسل ( Pixel-Level Segmentation) لتحسين عملية اكتشاف شقوق الرصيف وتوصيفها (التصنيف وتحديد الحجم)، وهو أمر حاسم من أجل سلامة البنية التحتية للطرق واستدامتها. بدأ البحث بجمع بيانات مكثفة، باستخدام كاميرا (GoPro Hero 8) مثبتة على مركبة لجمع صور الطرق في كوالالمبور وسيلانجور، في ماليزيا. وبعد إجراء معالجة مسبقة مفصّلة، شملت تحسين الصورة، وتشويشها، وضبط شدة الإضاءة، تمّ إعداد مجموعة البيانات الخاصة بشقوق الرصيف للتحليل. بدأت الدراسة بنموذج (YOLOv7) مخصص، حيث حقق نسبة استدعاء قدرها 0.9545 ودقة قدرها 0.9523 على مجموعة بيانات (RDD2022)، ونسبة استدعاء قدرها 0.9158 ودقة قدرها 0.93 على مجموعة البيانات المخصصة التي لدينا. وعند مقارنته بالطرق التقليدية، مثل الشبكات العصبية الالتفافية (ConvNets) والشبكات العصبية العميقة (Deep Neural Networks)، أظهر نموذج (YOLOv7) المخصص أداءً أفضل من حيث الدقة والاسترجاع. وأسفر العمل اللاحق مع نموذج (YOLOv8x-seg) عن تحسين الدقة والاستدعاء في اكتشاف الشقوق، بمقاييس أداء بلغت 0.93 للاستدعاء و0.91 للدقة، للشقوق التمساحية، و0.95 للاستدعاء و0. 84 للدقة، للشقوق الطولية، و0.806 للاستدعاء و0.89 للدقة، للشقوق العرضية. وبالمقارنة المباشرة، يتفوق هذا النموذج على الطرق السابقة التي تستخدم الشبكات العصبية الالتفافية العميقة المعتمدة على الدمج، من حيث الدقة والاسترجاع. وتضمنت المرحلة النهائية إنشاء نموذج هجين متقدم للتعلم العميق، يدمج بين الشبكة المتبقية ذات التدرج العميق (Deep Gradient ResNet) وآلية الانتباه المعدل ( Modified Attention)، وهو ما أدّى إلى تحسين عملية اكتشاف الشقوق وتوصيفها. وقد أظهر هذا النموذج دقة واعدة بلغت 0.84 للشقوق التمساحية، و0.89 للشقوق الطولية، و0.87 للشقوق العرضية، مع معدلات استرجاع بلغت 0.96 للشقوق التمساحية، و0.88 للشقوق الطولية، و0.80 للشقوق

العرضية. وبذلك، يتفوق النموذج المطوّر على البحث المقارن به، والذي يستخدم نموذج (CrackNet)، في كلّ من الدقة والاسترجاع. أثبتت الدراسة أنّ استخدام تقنيات التجزئة المتقدمة (Advanced Segmentation) والرؤية الحاسوبية (Machine Vision)، له القدرة على تحديد حجم شقوق الرصيف بدقة، وهو أمر حيوي لتحقيق الصيانة المستهدفة. كما تشير الدراسة إلى تقدم في التحليل الآلي لشقوق الرصيف، الأمر الذي يسهم في إنشاء بنى تحتية للطرق أكثر أماناً ومتانة.

# APPROVAL PAGE

The thesis of Arselan Ashraf has been approved by the following:

Ali Sophian
Supervisor

Teddy Surya Gunawan
Co-supervisor

Internal Examiner

Internal Examiner

External Examiner

Chairman

## DECLARATION

I hereby declare that this thesis is the result of my own investigations, except where otherwise stated. I also declare that it has not been previously or concurrently submitted as a whole for any other degrees at IIUM or other institutions.

Arselan Ashraf

Signature:  Date: 22-09-2024

**INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA**

**DECLARATION OF COPYRIGHT AND AFFIRMATION OF FAIR USE OF UNPUBLISHED RESEARCH**

**PAVEMENT CRACK DETECTION AND CHARACTERIZATION USING DEEP LEARNING AND PIXEL LEVEL SEGMENTATION**

I declare that the copyright holder of this thesis are jointly owned by the student and IIUM.



By signing this form, I acknowledged that I have read and understand the IIUM Intellectual Property Right and Commercialization policy.

Affirmed by Student Name

…… 22-09-2024 ……..

Signature Date

## ACKNOWLEDGEMENTS

All praise and gratitude are due to Allah, the Most Gracious and Merciful, for His endless blessings and guidance throughout my academic journey. It is with His grace that I have found the strength and perseverance to navigate the challenges of this program and complete my thesis.

I extend my heartfelt appreciation to my beloved parents. Their unwavering support, love, and prayers have been the foundation upon which I have built my dreams. Their sacrifices and belief in me have been my guiding light, providing me with the courage and resilience to pursue my academic goals.

I am profoundly indebted to my main supervisor, Assoc. Prof. Dr. Ali Sophian, for his invaluable mentorship, patience, support, and expertise. His insightful feedback and dedication have greatly enhanced the quality of my research.

Special acknowledgment goes to my co-supervisors, especially Prof. Teddy Surya Gunawan, whose guidance and collaborative spirit have significantly contributed to the direction and outcome of this work.

I also wish to express my sincere thanks to the Kulliyyah of Engineering at the International Islamic University Malaysia for providing me with the Kulliyyah of Engineering Merit Scholarship. This generous support has been instrumental in my academic pursuits.

I am grateful to my beloved university, International Islamic University Malaysia, for fostering an environment conducive to learning and personal growth.

Once again, I am humbly thankful to Almighty Allah for His infinite mercy and blessings which have made this achievement possible. Alhamdulillah.

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# LIST OF ABBREVATIONS

| | |
|---|---|
| 3D | Three-Dimensional |
| AI | Artificial Intelligence |
| ASPS | Augmented SubPixel Shuffling |
| AUC | ROC Area Under the Receiver Operating Characteristic Curve |
| BERT | Bidirectional Encoder Representations from Transformers |
| CNN | Convolutional Neural Networks |
| CPMC | Constrained Parametric Min-Cuts |
| CT | Computed Tomography |
| DL | Deep Learning |
| DQNs | Deep Q-Networks |
| E-ELAN | Extended Efficient Layer Aggregation |
| FN | False Negative |
| FP | False Positive |
| GANs | Generative Adversarial Networks |
| GLCM | Gray-Level Co-occurrence Matrix |
| GPS | Global Positioning System |
| GRU | Gated Recurrent Unit |
| HCI | Human-Computer Interaction |
| HOG | Histogram of Oriented Gradients |
| HSV | Hue, Saturation, Value |
| IDE | Integrated Development Environment |
| IOU | Intersection Over Union |
| IoT | Internet of Things |
| LSTM | Long Short-Term Memory |
| MAE | Mean Absolute Error |
| MAP | Mean Average Precision |
| MAPE | Mean Absolute Percentage Error |
| MCC | Matthews Correlation Coefficient |
| MSE | Mean Squared Error |
| MSLE | Mean Squared Log Error |

| | |
|---|---|
| NLP | Natural Language Processing |
| NMS | Non-Maximum Suppression |
| OLS | Ordinary Least Squares |
| PAM | Partitioning Around Medoids |
| PCA | Principal Component Analysis |
| PPO | Proximal Policy Optimization |
| RBF | Radial Basis Function |
| RDD | Road Damage Dataset |
| ReLU | Rectified Linear Unit |
| ResNet | Residual Network |
| RGM | Region Growing Method |
| RGB | Red, Green, Blue |
| RL | Reinforcement Learning |
| RMSE | Root Mean Squared Error |
| RNN | Recurrent Neural Network |
| SSD | Single-Shot Detector |
| SPP | Spatial Pyramid Pooling |
| SVM | Support Vector Machines |
| TN | True Negative |
| TP | True Positive |
| VAEs | Variational Autoencoders |
| YOLO | You Only Look Once |

# LIST OF SYMBOLS

| | |
|---|---|
| $y$ | Dependent Variable |
| $\beta 0$ | Bias Term |
| $\beta 1, \beta 2, \ldots, \beta n$ | Coefficients of the Independent Variables $x1, x2, \ldots, xn$ |
| $\varepsilon$ | Error Term |
| $e$ | Base of the Natural Logarithm |
| $Wmn$ | Weight of the kernel at position (m,n) |
| $b$ | *bias* |
| $Pij$ | Pooled value at position (i,j) in the pooled feature map |
| $ht$ | Hidden state |
| $Wih$ | Weight matrix between the input and hidden layer |
| $Whh$ | Weight matrix between the hidden layer and itself |
| $bh$ | Bias term for the hidden layer |
| $\sigma$ | Sigmoid activation function |
| $\odot$ | Element-wise multiplication |
| G | Generator |
| D | Discriminator |
| $z$ | Random noise vector |
| $A$ | Attention weights |
| MAE | Mean Absolute Error |
| MSE | Mean Squared Error |
| R2 | R-Squared |
| MAP | Mean Average Precision |
| lr0 | Initial Learning Rate |
| lrf | Final Learning Rate |
| cls | Class loss gain |
| dfl | Dual Focal Loss |
| kobj | Keypoint Object Loss Gain |
| $C_p$ | Crack Pixels |
| $T_{wp}$ | Total Number of White Pixels in the Binary Image |
| $C_l$ | Crack Length |

| | |
|---|---|
| $F(x)$ | Residual mapping |
| $H(x)$ | Underlying mapping |
| $\mu_B$ | Batch mean |
| $\sigma_B^2$ | Batch variance |
| $N$ | Number of samples |
| $\hat{y}$ | Predicted label |
| $\approx$ | Approximation |
| ~ | Weak Approximation |
| + | Addition operator |
| - | Subtraction operator |
| % | Percentage |

# CHAPTER ONE

# INTRODUCTION

## 1.1 BACKGROUND OF THE STUDY

Cracks in the road surface significantly impact pavement performance. Timely and precise identification of these deteriorations is essential for effective pavement upkeep. Such cracks not only lead to various forms of road damage but also compromise the aesthetic and comfort of the driving experience. Furthermore, they can undermine the structural strength of the pavement, leading to a reduction in its overall lifespan (Ha et al., 2022). With an increase in vehicular traffic, the emphasis on pavement conservation has heightened. Consequently, proactive measures in detecting and repairing pavement cracks can substantially reduce repair expenses while enhancing road safety for both vehicles and their operators.

Currently, most initial approaches to identifying and addressing road cracks depend on manual methods. These methods are not only time-consuming and challenging but also suffer from low efficiency and inherent safety risks (Munawar et al., 2021). Ensuring that pavements remain in good condition is crucial for the safety of motorists, a responsibility held by both local and state road authorities. A key task in this duty involves assessing pavement damage, a process that is not only painstaking but also demands specialized skills. However, with the swift progress in artificial intelligence, particularly convolutional neural networks, there has been a significant impact on image recognition capabilities (S. Zhou et al., 2016). In recent times, deep learning methods have gained popularity for their application in identifying and classifying road surface cracks (Q. Zou et al., 2019). The integration of deep learning with road crack detection technologies has markedly enhanced the efficiency and accuracy of crack identification. Moreover, the use of computer vision and Artificial Intelligence (AI) has proven effective in automating the process of inspecting pavement cracks, further advancing the field (Jahangiri & Rakha, 2015) (Hu et al., 2021).

Researchers have conducted extensive studies on detecting pavement cracks, proposing a wide range of solutions from image processing to advanced AI and deep learning techniques, which are increasingly popular today. Within the realm of image processing, three primary approaches are highlighted (Kamdi & Krishna, 2012), which are; edge segmentation, edge detection, and region detection. Edge segmentation involves categorizing image pixels into distinct groups by setting a specific intensity threshold to distinguish the target crack from its surroundings. Edge detection methods identify the contours of cracks using algorithms like Sobel, Prewitt, and Canny detectors. Meanwhile, region detection focuses on identifying areas within the crack by clustering pixels with similar characteristics. The challenge of automating pavement crack detection from images has been a longstanding issue in traffic engineering and pattern recognition, attracting significant attention from the research community (Ranyal et al., 2024). The detection of road damage is often difficult because of the multi-surface, multi-level design, the feeble sign of targets, and the variation of pixel intensity. The majority of the methods are built with high image quality and clear crack targets in consideration, but they lack the flexibility to adapt to complicated environments, making it difficult to achieve the functional requirements. The gathering of road data is at the core of the investigation into automated road crack recognition. Considering the importance and need for an automated and accurate road crack detection and characterization (classification and sizing) model, this research has been conducted.

## 1.2 PROBLEM STATEMENT

Conventional methods for detecting and assessing road cracks rely heavily on manual inspections, which are prone to subjectivity, time-consuming, costly, and present potential hazards to workers. Additionally, these methods demand significant labor resources, making them impractical for large-scale applications. While machine-vision systems have emerged as a promising alternative, current algorithms for crack detection and characterization face critical challenges in terms of accuracy, recall and precision. A key issue is their reliance on local analysis, which often results in overfitting and poor performance in complex environments.

Another significant challenge is the computational intensity required for classifying crack regions. Processing large volumes of image data is resource-intensive, and background noise further complicates the task, reducing system overall practicality. Most existing approaches also focus on block-level crack identification, limiting the precision of crack sizing and detailed assessments. Pixel-level analysis has been suggested as a more accurate alternative, but its potential in crack detection and characterization remains underexplored.

Many existing research studies primarily focus on training models using clean and pre-processed crack images, which simplifies the detection process in controlled environments. However, this approach often overlooks the inclusion of noise, such as shadows, pixel intensity variations, and different types of noise like Gaussian and salt-and-pepper, that are prevalent in real-world conditions. By not incorporating noisy data during training, these models tend to perform well in ideal scenarios but fail to generalize effectively when faced with the complexities of real-world environments. To ensure models are more generalized and applicable to real-world conditions, it is crucial to train them on datasets that combine both clean and noisy data. This approach reflects the variability encountered in real scenarios and enhances the model's robustness and performance in diverse conditions.

Moreover, data collected from moving vehicles adds complexity, as repetitive frames of the same cracks can lead to inaccurate assessments. Large or elongated cracks that span multiple frames require advanced techniques to accurately measure their true dimensions.

## 1.3 HYPOTHESIS

This research hypothesizes that the application of Convolutional Neural Networks (CNNs) within deep learning frameworks can substantially elevate the accuracy of pavement crack detection and characterization. It posits that leveraging CNNs for their advanced pattern recognition capabilities can outperform traditional methods in identifying, categorizing and characterizing pavement cracks. Additionally, the hypothesis extends to the concept that analyzing extensive or long cracks through multiple video frames from a moving platform will enhance crack characterization,

offering a more dynamic and comprehensive view. Furthermore, the integration of pixel-level segmentation with deep learning techniques is expected to significantly refine the performance of crack characterization. This approach is likely to provide a detailed and precise analysis of pavement conditions, leading to more accurate and practical insights for pavement maintenance and repair initiatives.

## 1.4 RESEARCH QUESTIONS

1. Which CNN models can provide optimum results in characterizing cracks?
2. Can the proposed models based on CNN and pixel-level segmentation effectively detect, classify, and assess different types of cracks on the road?
3. How can images captured from a camera mounted on a moving vehicle be used to identify and characterize cracks on the road accurately, including large and long cracks that will be present in multiple consecutive frames?
4. Can the use of pixel-level segmentation lead to the more accurate sizing of the detected cracks?

## 1.5 RESEARCH OBJECTIVES

The aim of this research is to develop and evaluate robust deep learning models for automated pavement crack detection and characterization using a vehicle-mounted camera system. This is broken into the following objectives:

1. To identify suitable CNN models for pavement crack detection and characterization.
2. To develop an automated pavement crack detection database via vehicle-mounted camera setup to improve road maintenance analytics.
3. To develop robust deep learning models that integrates refined object detection with pixel-level segmentation techniques for the accurate detection and characterization of pavement cracks.

4. To evaluate the performance of the developed models.

## 1.6 RESEARCH SCOPE

This study focuses on developing a machine vision-based model specifically tailored for the detection and characterization of road cracks, using image and video-based datasets exclusively. Other data modalities, such as sensor or geospatial data, fall outside the scope of this research to ensure a concentrated effort on visual data processing. The model employs pixel-level segmentation for crack detection, chosen for its precision in detailing crack patterns, while alternative segmentation methods are excluded to maintain consistency and focus.

The research is limited to the analysis of three main types of pavement cracks: longitudinal, transverse, and alligator (crocodile) cracks, selected due to their prevalence and significance in road infrastructure maintenance. The study utilizes the GoPro Hero 8 camera for data collection, chosen for its advanced image stabilization, ensuring clear and stable footage from moving vehicles. Cracking on surfaces other than pavements, such as sidewalks or bridges, is not considered in this study. By narrowing the scope in this way, the study is able to produce more focused and reliable insights into pavement crack detection and characterization.

## 1.7 RESEARCH METHODOLOGY

To achieve the objectives of this research, the following work will be carried out as shown in Figure 1.1.

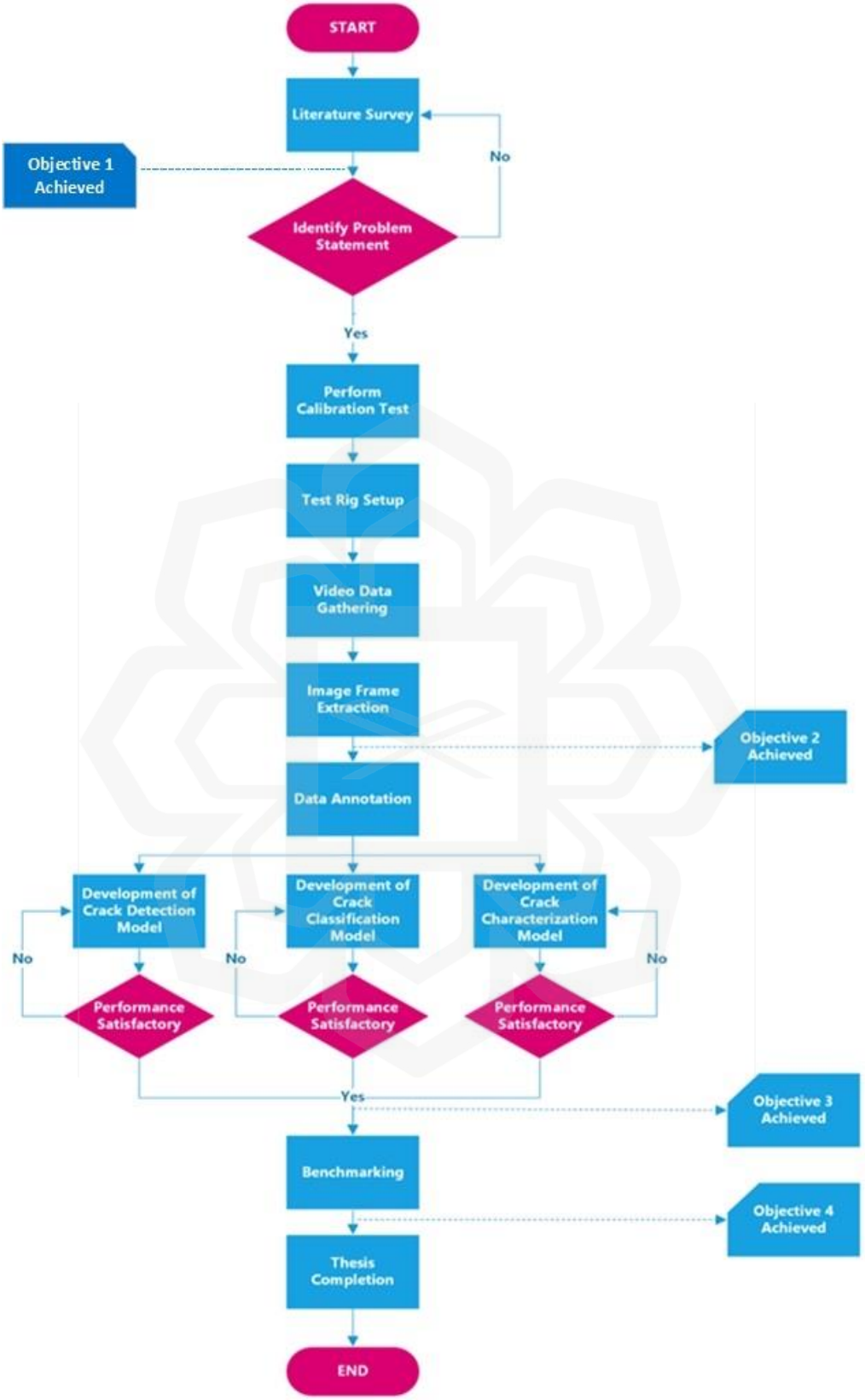



Figure 1.1 Pavement crack detection and characterization using deep learning and pixel level segmentation.

1. A literature survey will initially be conducted to identify suitable models for detection, classification and characterization of cracks found in pavements along with problems associated with them.
2. The second step of the process is the calibration test. The height of the camera will be adjusted so that it will cover a width of 3.1 m will be calculated through the calibration test. The width of 3.1 m is based on the maximum width of standard roads in various countries.
3. The next step is the data gathering. Both publicly available and purposely collected data will be obtained. Publicly available data include RDD datasets, Crack500, or more. For local road data gathering, a camera mounted at the back or in front of a moving vehicle, which, in turn, allow the recordings of videos containing cracks. Suitable cameras, likely to be a GoPro Hero 8 camera that has stabilization feature, will be used. GoPro cameras are known for their video quality and structural ruggedness. Recordings of videos containing cracks in selected roads around International Islamic University, Kuala Lumpur and Selangor will be obtained.
4. The next step is the development of the preprocessing stage which will be applied to the input image or video sample to improve the detection, classification and characterization. In case of video input, the frames will be extracted. The number of frames to be extracted will be determined based on complexity and time. The obtained frames will be further color corrected if the need arises. This step will be followed by the histogram equalization of frames. Histogram equalization is a computer-based image processing strategy used to improve contrast in pictures. Data labelling will be done for the training and validation data.
5. The next step is the development of the crack detection by using pixel level segmentation and deep learning model like YOLO, Deep CNN, ResNet, etc. because it can train several layers without increasing training error and tackles with vanishing gradients. Based on the literature review, a model will be decided, built and assessed. It is very important to detect the crack from the image or video frame for further processes.
6. Following the crack detection, the classification and characterization of

the cracks will be performed based on type and size respectively. An effective technique of using multiple consecutive frames will be developed in order to characterize large cracks that appear across multiple frames.

7. The crack detection and characterization model will be trained and validated by using the datasets gathered at the beginning of this research project.
8. Finally, the performance of the developed model will be evaluated.

## 1.8 RESEARCH SIGNIFICANCE

The anticipated results of this research hold substantial potential in revolutionizing the approach towards road maintenance, particularly in the detection and characterization of road cracks. Traditional methods employed for road crack detection and assessment, which are predominantly manual, are often marred by subjectivity, time-intensive processes, potential risks to workers, and considerable labor demands. The introduction of a machine vision-based approach in this research promises to address and mitigate these limitations effectively.

A cornerstone of this study is the use of pixel-level segmentation, a technique poised to significantly elevate the accuracy and overall performance of crack detection systems. This advanced method enables a more precise and detailed analysis of cracks, surpassing the capabilities of conventional approaches. By delivering more accurate and reliable results, this research aims to substantially improve the efficiency and safety of road maintenance operations.

Furthermore, this research is not just about the immediate technological advancements it brings; it's also about inspiring future exploration in this field. By demonstrating the efficacy and benefits of machine vision in real-time road crack detection, this study seeks to pave the way for further investigations and innovations. It aims to serve as a catalyst, encouraging more researchers to delve into this domain and contribute to the ongoing evolution of methods and technologies for road infrastructure maintenance and safety.

## 1.9 THESIS ORGANIZATION

This remaining of the thesis is structured as follows: Chapter 2 provides a comprehensive review of the literature, delving into studies related to the detection and characterization of pavement cracks. Chapter 3 outlines the research methodology and implementation details. Chapter 4 is dedicated to the presentation and discussion of the results. The dissertation concludes with Chapter 5, which offers a summary of the conclusions drawn and suggests directions for future research.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1 INTRODUCTION

Annually, a significant amount of funding is allocated to procure various tools and technologies aimed at identifying damages within crucial infrastructures like roads, bridges, and buildings (Ni et al., 2019). Continuous exposure to environmental elements like sunlight and precipitation, along with geological movements and natural aging, significantly impacts the durability of civil infrastructure such as roads, bridges, and pavements. These factors collectively affect their functionality and longevity in various ways (Tran et al., 2021). These events can lead to severe structural failures or manifest as physical damages, often in the form of cracks. Such cracks typically start on a very small scale (Liong et al., 2019) but can compromise the pavement's structural integrity, reduce its ability to bear loads, and create surface irregularities (Dung & Anh, 2019). Early detection of these cracks can prevent further deterioration. Neglected cracks may expand, shortening the pavement's service life, and potentially causing accidents, injuries, and financial losses.

Traditional approaches to early detection and classification of pavement cracks primarily depend on manual methods. These involve skilled personnel conducting visual inspections and utilizing specialized tools to detect any defects (Oliveira & Correia, 2013). However, these manual strategies are known for being both time-consuming and labor-intensive, with limitations in detection precision and inherent risks (Ashraf, Sophian, et al., 2022a). In recent years, image-based algorithms for crack detection have gained prominence. The early stages of this research trend were marked by efforts to refine or merge traditional digital image processing approaches, including but not limited to mathematical morphology, thresholding techniques, and methods for detecting edges (Decker, 2016). These strategies leverage the photometric and geometric attributes inherent in images of cracks (X. Yang et al., 2018), with a particular emphasis on the darker appearance of crack pixels. This characteristic has been instrumental in developing thresholds, both global and local, to distinguish

cracks from their surroundings. However, such pixel-focused techniques are often challenged by the presence of noise. To counteract this, some strategies have been introduced that utilize geometric information, such as the principle of crack linearity to minimize false detections, or the application of the local binary pattern operator to assess a pixel's crack-relatedness based on its directional properties (Sivamayil et al., 2023). Furthermore, multiscale approaches utilizing wavelet transformations are employed to differentiate cracked from intact areas, enhancing the ability to identify cracks, albeit with limitations in capturing every single crack in an image.

The evolution of Artificial Intelligence (AI) and advancements in computer vision have catalyzed a wave of innovative solutions for the automated detection of cracks (L. Zhang et al., 2016). AI has been widely recognized for its utility in tackling an array of complex issues (Ghimire, Thapa, Jha, Kumar, et al., 2020) extending its influence across various sectors, including but not limited to finance and healthcare. This has cemented the role of AI and machine vision as pivotal elements across numerous disciplines (Ghimire, Thapa, Jha, Adhikari, et al., 2020). The integration of machine learning, particularly deep learning, into both industry and academia has positioned deep learning algorithms as potent resources for the autonomous identification and categorization of pavement cracks (Ma et al., 2020). Since the inception of machine learning, a plethora of methodologies focusing on feature extraction and pattern recognition have been developed for the purpose of crack detection (Shi et al., 2016a). While these techniques have shown impressive efficacy, their performance is largely contingent upon the quality of the extracted features. The heterogeneity in pavement conditions poses a significant challenge in identifying features that are universally effective. Nonetheless, deep learning techniques have demonstrated significant potential in overcoming these hurdles in the field of roadway maintenance, igniting a growing interest in research within this domain. This heightened research activity is underscored by the increasing volume of related studies, as depicted in Figure 2.1.

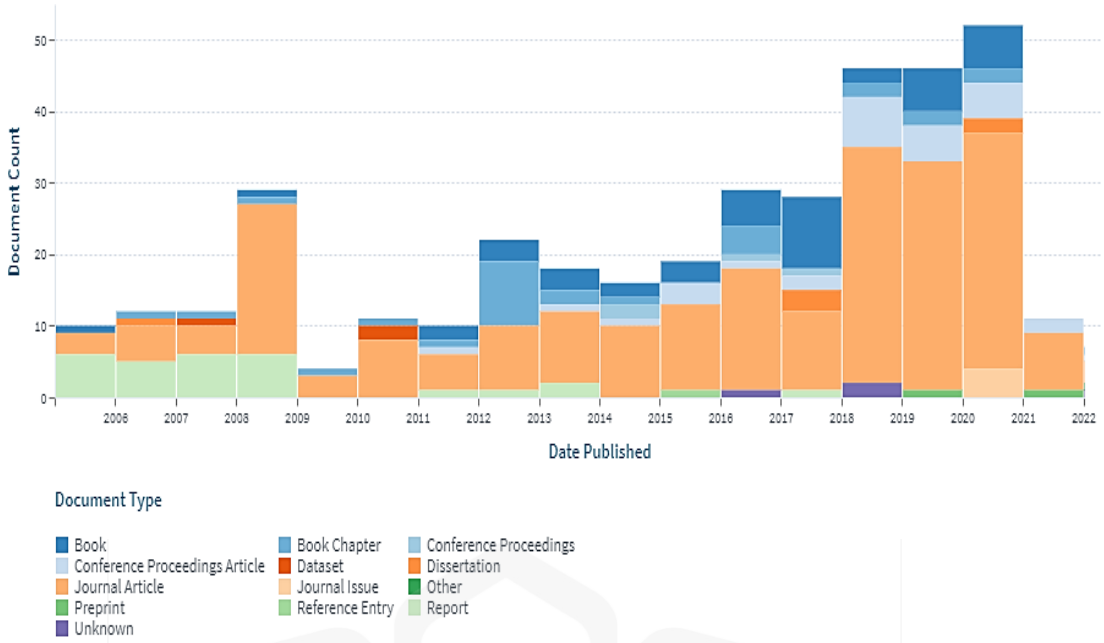



Figure 2.1 Evolution of Research in Deep Learning for Pavement Crack Detection and Classification *(Accessed from Lens.org on 03/03/2022)*

## 2.2. PAVEMENT CRACKS

Cracks in pavement surfaces primarily impact the uppermost layer, leading to a range of surface-level problems on roads. Such damage generally begins as the pavement ages and bears the weight of regular traffic. With time, these cracks can allow water to seep in, aggravating the damage and potentially leading to more significant issues like potholes. Damage to road surfaces not only causes inconvenience to users but can also become safety hazards if left unattended, with the potential for the situation to deteriorate further over time. It is crucial to undertake timely and effective maintenance to extend the lifespan of roads and minimize the costs associated with their repair and upkeep. Although road assessments often take into account factors such as surface attributes (including roughness and the longitudinal profile), the state of the asphalt, preliminary evaluations, and the quality beneath the surface, there currently exists no uniform approach for the collection of data on pavement conditions. Each road management entity follows its unique set of guidelines (Zeiada et al., 2019), leading to a variety of strategies for addressing similar issues (Pantuso et al., 2019).

### 2.2.1. Pavement Crack Types

Pavements primarily fall into two categories: concrete and bituminous (or asphalt). Understanding the specific nature of a pavement crack is essential before implementing the correct repair strategy. This step ensures the restoration method is tailored to the pavement's material and the particular characteristics of the crack, leading to more effective and durable repairs.

#### *2.2.1.1. Concrete pavement Cracks*

Concrete pavements, often referred to as rigid pavements, constitute a solid layer that directly interfaces with vehicular traffic and serves a multitude of functions. Figure 2.2 showcases the various crack patterns typically found in concrete pavements, including shrinkage cracks, transverse cracks, longitudinal cracks, block cracks, diagonal cracks, and corner cracks.

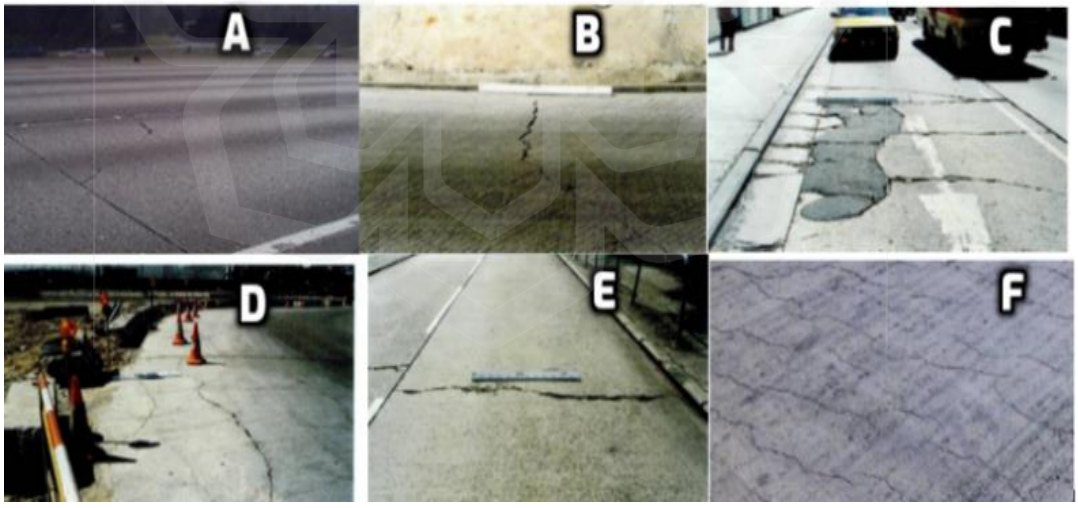



Figure 2.2. Classification of Cracks in Concrete Pavements: (A) Corner, (B) Diagonal, (C) Block, (D) Longitudinal, (E) Transverse, (F) Shrinkage

A corner crack originates from the slab's corner, extending towards the joints and not exceeding half the slab's length on either side. Diagonal cracks, characterized

by their multi-directional pattern, are neither perpendicular nor parallel to the pavement's length and typically do not extend to inlets' boundaries. Block cracks form a network of interlinked rectangles that evenly cover the asphalt surface. Longitudinal cracks run parallel to the direction of the pavement, presenting as discontinuous lines. Transverse cracks cut across the pavement width, perpendicular to its length. Shrinkage cracks, often short and diagonal, usually stop short of reaching the pavement edges.

#### *2.2.1.2. Bituminous Pavement Cracks*

Figure 2.3 displays the typical crack formations in bituminous pavements, which are characterized by their asphaltic surface layers originating from dense, viscous crude petroleum. The prevalent crack patterns in these pavements encompass alligator cracks, slippage cracks, transverse cracks, longitudinal cracks, block cracks, and diagonal cracks.

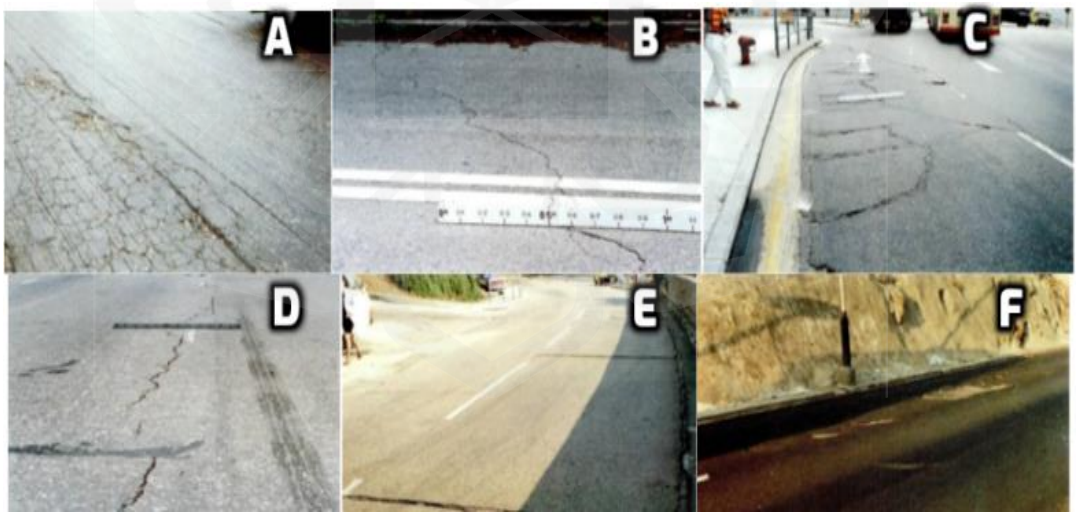



Figure 2.3. Common Crack Patterns in Bituminous Pavements: (A) Alligator, (B) Diagonal, (C) Block, (D) Longitudinal, (E) Transverse, (F) Slippage

Alligator cracks resemble an alligator's skin with a network of interlocking lines forming small polygonal shapes across the pavement. Diagonal cracks are isolated lines that cut across the asphalt at an angle, similar to a diagonal line on a

grid. Block cracks create a pattern of large, interconnected cells reminiscent of a grid or a series of blocks. Longitudinal cracks run parallel to the road's direction, creating lines that follow the length of the pavement. Transverse cracks appear as single, distinct lines that cut across the pavement surface at a right angle to the direction of the road. Slippage cracks, shaped like crescents, curve away from the flow of traffic, indicating movement of the asphalt layer.

### 2.2.2. Assessing Pavement Crack Characteristics

The analysis of pavement conditions can be enhanced through the automated measurement of crack features, including length, width, area, and depth (Y. Zhao et al., 2023a). Crack evaluation often depends on the type of crack, with crack length being the primary metric for longitudinal and transverse cracks, and crack area being crucial for assessing block and alligator cracks.

Traditionally, inspectors relied on tools like strings, graded scales, and crack comparators for manual measurements. However, these methods suffered from low accuracy, subjective interpretations, and difficulties in data recording (Gehri et al., 2020). The digital pachymeter, offering higher resolution, also presented challenges, requiring precise insertion of a metallic blade into the crack by a skilled technician. The inherent subjectivity in manual placement and the risk of inaccuracies arising from technician intervention often required multiple measurements to enhance precision (Koch et al., 2015), a process deemed inefficient regarding expenses, time, and labor. However, advancements in machine vision and machine learning technologies are progressively addressing these challenges, facilitating the automated measurement of crack parameters.

## 2.3. PAVEMENT CRACK IDENTIFICATION AND ANALYSIS FRAMEWORK

Enhancements in computing power have paved the way for creating advanced systems in image processing and machine learning, specifically designed for identifying and

categorizing pavement cracks. This area has seen considerable growth, with researchers employing a range of techniques and processes. Although approaches vary, the underlying structure of systems designed for crack detection and categorization remains largely uniform. Such systems are built by seamlessly combining different components, resulting in a robust mechanism that can efficiently detect, categorize, and analyze pavement cracks, as shown in Figure 2.4, where the system is broken down into separate modules.

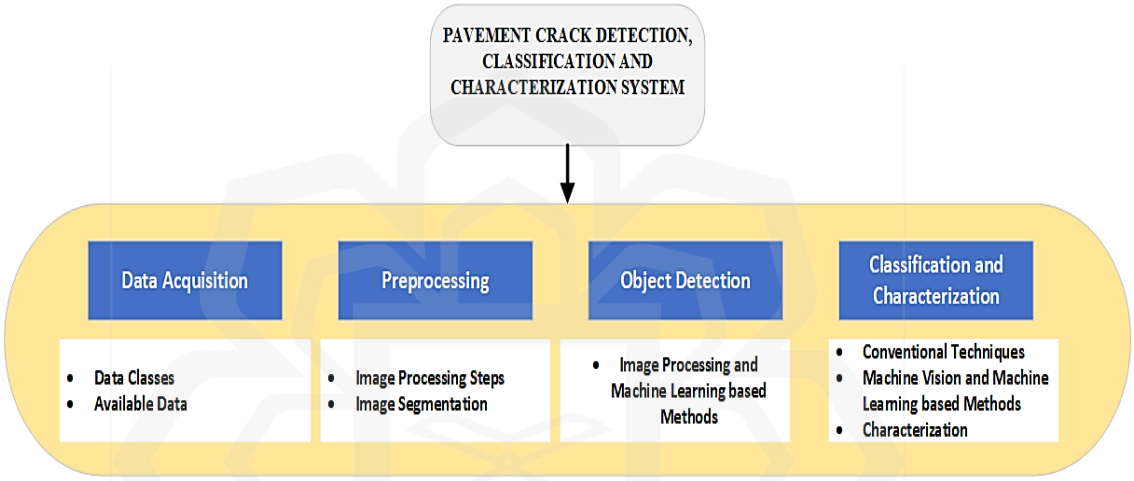



Figure 2.4. Integrated Approach to Pavement Crack Analysis

At the core of any model designed for detecting objects, the initial step involves gathering input data, typically sourced from images or video footage. Researchers often draw upon pre-existing data collections to establish their frameworks, further enriching these foundations with bespoke datasets tailored to their specific research needs. These tailored collections are assembled through a range of data gathering techniques, which might include the use of smartphones, digital cameras, satellite images, or laser scanning technologies. After acquiring the data, it undergoes a critical phase of pre-processing, during which the raw inputs are subjected to a variety of image enhancement methods. This stage could entail tasks such as image cropping, resizing, enhancing contrast through histogram equalization, and adjusting colors to improve the visual quality of the input. The result is a polished, ready-for-analysis dataset that feeds into the algorithms responsible for identifying and categorizing cracks.

### 2.3.1. Data Acquisition for Crack Analysis

Gathering data plays a pivotal role in creating machine learning models, much like oil fuels the modern economy. The effectiveness of machine learning models greatly depends on the quality of the training data used (Ashraf et al., 2021), (Ashraf, Gunawan, et al., 2022). For pavement crack detection and characterization systems, it is essential to gather exclusive and significant data. Researchers typically use images or videos as the primary data format for developing machine learning-based road crack detection systems. This section examines various classes of image acquisition data relevant to road crack detection along with various available databases.

#### *2.3.1.1. Categories of Crack Images*

Images of cracks are commonly sorted into six unique types, reflecting their visual characteristics as documented in numerous scholarly investigations. Table 2.1 outlines these various categories of images.

Table 2.1. Types of Crack Images

| Image Type | Applicability |
|---|---|
| Notable light pictures | These images, captured using magnification devices, ground and aerial cameras, as well as satellites, exhibit exceptional clarity and quality. |
| Laser based | Primarily utilized for image processing and analyzing 3D cracks, typically ranging from a few millimeters to several meters in diameter. |
| CT based | Employed in laboratory settings to examine microscopic cracks, varying in size from micrometers to millimeters. |
| Radar images | Used for assessing crack depth and generating satellite imagery. |
| Ultrasonic images | Utilized for wavelet and curvelet transformations. |
| Infrared images | Employed to gauge crack depth, especially in unique circumstances. |

Merging multiple classes of image data offers significant benefits for the detection and evaluation of cracks. By amalgamating images from various sources and utilizing this combined data, there's potential to develop a more holistic, precise, and inclusive visual depiction for the extraction of scene details. This approach opens up a broad spectrum of possible uses.

#### *2.3.1.2. Road Crack Datasets*

The success of a machine learning model is deeply dependent on the caliber of data it is trained on. Hence, the selection of a suitable dataset is vital (Ashraf et al., 2020). Raw data alone is insufficient for training and testing machine learning algorithms; it necessitates preprocessing to align with the demands of particular algorithms. For the purpose of constructing models capable of identifying and categorizing road cracks, scholars have employed diverse datasets related to pavement cracks. Table 2.2 offers a synopsis of the datasets integrated into this structure.

Table 2.2. List of datasets for Pavement Crack Analysis

| **Dataset** | **Image Sample Count** | **Image Resolution** | **Image Format** | **Data Enhancement** |
|---|---|---|---|---|
| Concrete Crack Images for Classification (Özgenel & Sorguç, 2018) | 40000 | 227 x 227 pixels | Colour | Not Applicable |
| RDD2020: An Image Dataset for Smartphone-based Road Damage Detection and Classification (Arya et al., 2021) | 26336 | 600 x 600 pixels | Colour | Not Applicable |
| Asphalt Crack Dataset (Jayanth Balaji et al., 2018) | 400 | 448 x 448 pixels | Colour | Not Applicable |
| EdmCrack600: a Pixel-level annotated dataset for crack identification (Mei et al., 2020) | 600 | 256 x 256 pixels | Colour | Not Applicable |

| | | | | |
|---|---|---|---|---|
| Crack Detection: Image Classification | 15168 | Not Available | Colour | Not Applicable |
| Cracks-and-Potholes-in-Road-Images-Dataset (B. T. Passos et al., 2020) | 2235 | 2560 x 1440 pixels | Colour | Not Applicable |
| CFD (Shi et al., 2016b) | 118 | 320 x 480 pixels | Colour | Not Applicable |
| AigleRN database (Chambon & Moliard, 2011) | 38 | Not Available | Greyscale | Not Applicable |
| CRACK500 | 500 | 2000 x 1500 pixels | Colour | Applicable |
| German Asphalt Pavement Distress (GAPs) Dataset (Eisenbach et al., 2017) | 1969 | 1920 x 1080 pixels | Greyscale | Not Applicable |
| Cracktree200 (Naik & Murthy, 2003) | 206 | 800 x 600 pixels | Colour | Not Applicable |

### 2.3.2. Image Pre-Processing

Image preprocessing is the technique of performing initial computations and transformations on raw image data to prepare it for further analysis. These transformations often enhance the image quality, reduce noise, highlight essential features, or extract specific attributes required for the task at hand (Mohan & Poobal, 2018a). Importance of Image Preprocessing are listed as under:

- Improving Accuracy: By reducing the unwanted variations in images, preprocessing can help the machine learning model to focus on the relevant patterns, thereby increasing its predictive accuracy.
- Speeding up Computation: Proper preprocessing can reduce the complexity of the images, leading to faster computations and a more efficient learning process.
- Enhancing Generalization: Preprocessing helps in creating consistent and standardized data, ensuring that the model can generalize well across various types of images.

- Reduction of Noise: Images often contain noise or irrelevant information. Preprocessing can mitigate these factors, resulting in cleaner data.

#### *2.3.2.1. Different Image Preprocessing Steps*

1. Grayscale Conversion: This process involves converting a colored image into shades of gray. The main reason for this transformation is to simplify the image information, reduce computational load, and, in some cases, improve algorithm performance (Zeger & Grgic, 2020). A grayscale image retains only intensity information and removes color information, which isn't always necessary for certain tasks, such as document digitization, shape detection, etc.
2. Histogram Equalization: A histogram of an image represents the distribution of pixel intensities. Histogram equalization aims to adjust the image to have a uniform histogram, thus improving the contrast and enhancing hidden features. This method is particularly useful in scenarios where the image has poor contrast due to lighting conditions (Mustafa & Abdul Kader, 2018).
3. Noise Reduction: Digital images are often subject to various types of noise, which may hinder image analysis (Uzakkyzy et al., 2023). Several techniques are available to reduce noise. For example, Gaussian blurring uses a Gaussian filter to convolve with the image. This process smoothens the image and reduces detail and noise. Median filtering, on the other hand, replaces each pixel value with the median value of its neighbors, which is particularly effective against 'salt-and-pepper' noise.
4. Normalization and Standardization: Normalization typically involves scaling pixel intensity values to a range between 0 and 1, which can simplify the computational operations and help certain algorithms converge faster. Standardization goes a step further by ensuring that the pixel values follow a standard normal distribution with a mean of 0 and a standard deviation of 1. This is particularly useful when working with deep learning models (Muhammad Ali & Faraj, 2014).
5. Edge Detection: Edge detection is the process of finding points in an

image where the brightness changes sharply, indicating the boundaries of objects within the scene (Ganesan & Sajiv, 2017). The Sobel operator is used to compute the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The Canny operator, another popular method, involves multiple stages including smoothing, computing a gradient, and non-maximum suppression to detect a wide range of edges in images.

6. Data Augmentation: Data augmentation involves creating new images by applying various transformations to the existing ones (Shorten & Khoshgoftaar, 2019). Techniques like rotation, scaling, translation, flipping, zooming, and cropping can generate new variations of images, thereby expanding the dataset. This process helps improve the performance of machine learning models by providing more diverse data and reducing overfitting.
7. Feature Extraction: Feature extraction involves transforming the raw image data into a set of features or attributes that are more meaningful and informative (Kumar & Bhatia, 2014). Techniques like Principal Component Analysis (PCA) reduce the dimensionality of the data while preserving its variability. Other methods may involve extracting texture, color, or shape attributes. For example, Gabor filters can be used for texture extraction, color histograms for color features, and Hough transform for shape attributes.
8. Color Space Conversion: Color space refers to the method used to represent and organize color information. Images are often represented in RGB (Red, Green, Blue) color space, but certain tasks may require the conversion of these images into different color spaces such as HSV (Hue, Saturation, Value), or YCbCr (Luminance, Chrominance Blue, Chrominance Red), among others. These conversions can help emphasize certain aspects of the image that are more easily processed or identified in different color spaces (Bi & Cao, 2021).
9. Morphological Operations: Morphological transformations are operations based on the image shape employed on a binary image (Said & Jambek, 2021). Two of the most fundamental morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image,

while erosion removes pixels on object boundaries. The combination of these two operations can be used to remove noise, fill holes in objects, thin or thicken object boundaries, and more.

10. Resizing and Aspect Ratio Handling: Resizing is a crucial step when your machine learning model expects a particular input image size. Additionally, while resizing, it's also essential to consider maintaining the aspect ratio of the image to prevent distortions. Resampling methods, such as bilinear and bicubic interpolation, are used during resizing to calculate the new pixel values (Ashraf et al., 2023).
11. Geometric Transformations: Geometric transformations, such as rotation, scaling, translation, and affine transformation (preserves lines and parallelism but not distances and angles), are often used in data augmentation to expand the dataset artificially and to achieve invariance to these transformations (Fang et al., 2023).
12. Image Pyramids: An image pyramid is a collection of scaled-down versions of an image, each one being a reduced resolution version of the previous one. They are used in image fusion, scale-invariant feature detection, and in creating a scale space, allowing to find features at different scales (S. Zhao & H. Ning, 2017).
13. Skeletonization: Skeletonization is a process to reduce foreground details in a binary image to represent a general form of an object (Abu-Ain et al., 2013). It works by thinning the foreground until only the "skeleton" of the original object remains.
14. Image Sharpening: Image sharpening enhances the edge contrast of an image, making it appear more defined (Lozano-Vázquez et al., 2022). It helps in increasing the detail of the image. The most commonly used technique is unsharp masking, which subtracts a blurred version of the image from the original image.
15. Texture Analysis: Texture analysis is the classification of regions based on their texture (Humeau-Heurtier, 2022). A common way of identifying texture features is using a Gray-Level Co-occurrence Matrix (GLCM) that measures the occurrences of specific pixel intensity levels occurring in an image.

### 2.3.3. Image Segmentation

Segmentation is the method involved with apportioning an image into numerous portions. Image segmentation is commonly used to find objects and limits in images. In an image grouping task, the organization appoints a mark (or class) to each information in image to know which pixel has a place with which object, and so forth. For this, a class is appointed to every image pixel. This undertaking is known as segmentation. A segmented model returns significantly more definite information about the image as shown in Figure 2.5. It has numerous applications in clinical imaging, self-driving vehicles and satellite imaging, for example. Deep learning methods have proven effective in distinguishing, outlining, and segmenting cracks (Mo et al., 2022) across a variety of materials including steel, concrete, asphalt and masonry (Chen et al., 2021). Generally, two approaches, namely classification and segmentation, have been utilized in crack detection literature. In the first approach, small patches of an image are labeled as either crack or non-crack. In the second approach, each pixel is classified as either crack or non-crack.

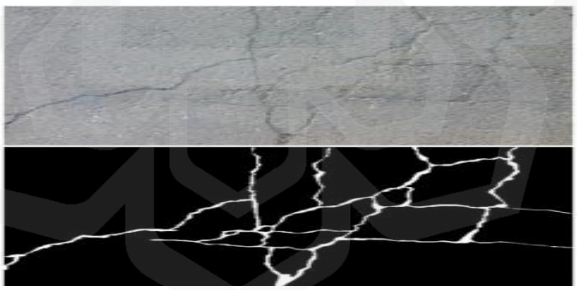


Figure 2.5. Crack Segmentation (P. Li et al., 2022a)

Image segmentation stands as a crucial step in processing images containing cracks. The purpose of image processing is to improve the clarity of crack-containing images, rendering the cracks more discernible (Yu et al., 2023). Pre-processing algorithms for crack images can be classified into global and local algorithms. The two most notable global calculations are difference stretches and histogram levelling in an image. Dynamic histogram adjustment, bit channels, dynamic histogram levelling, morphology, and multiscale preparation have a place in the neighbourhood

calculations. The global calculations modify all pixels in an image, while the resulting calculations use input-yield modifications based on neighbourhood attributes.

The amount and type of information conveyed by image segmentation tasks are divided into three categories: semantic, instance and panoptic (Abdulateef & Salman, 2021; Kar et al., 2021; X. Li & Chen, 2022). While semantic segmentation creates a broad limit of items belonging to a specific class, instance segmentation creates a segment map for each object it sees in the image, regardless of the object's class. Panoptic segmentation combines instance and semantic segmentation tasks, which is the most informative. The segment maps of all the items of any specific class present in the image are obtained using panoptic segmentation.

The categorization of pixels in a picture into semantic classifications is referred to as semantic segmentation. Pixels in each class are simply classified as belonging to that class, with no other information or context taken into account. When there are several instances of the same class in the image, it is a poorly articulated problem statement, as one might assume. Instead of classes, instance segmentation models divide pixels into groups based on "instances." The class of an identified region is unknown to an instance segmentation method, but it can separate overlapping or highly similar object sections based on their borders. The most recently developed segmentation technique, panoptic segmentation, can be described as a combination of semantic and instance segmentation. Each instance of an object in the image is separated, and the object's identity is anticipated.

Major image segmentation algorithms usually employed have been listed in Table 2.3, along with their advantages and limitations.

Table 2.3. Techniques for Image Segmentation

| Method | Detail | Pros | Cons |
|---|---|---|---|
| Edge Detection (P. Li et al., 2022b) | This approach identifies the boundaries within an image to highlight its features, as edges hold essential information and attributes. It streamlines the image by reducing | Works well with images where the contrast between | Ineffective for images plagued by significant noise. |

| | | | |
|---|---|---|---|
| | its size and discarding non-essential information, focusing on the key structural elements. | elements is pronounced. | |
| Histogram-Based Thresholding (Akagic et al., 2018) | Utilizes histogram peaks to group similar pixels, a fundamental step in distinguishing different parts of an image. | Simple to apply with minimal initial processing required. | Can miss crucial details and is sensitive to errors in setting the threshold. |
| Clustering Techniques (Talab et al., 2016) | These methods segment an image into k distinct, non-overlapping groups based on similarity, aiming to isolate objects within the image. | Reliable methods that can be improved with fuzzy logic, fitting for immediate processing tasks. | The complexity lies in defining an appropriate minimization cost function. |
| Region Segmentation (J. Liu et al., 2020) | This strategy divides an image into areas or segments that share common characteristics, often defined by pixel groupings. | Particularly useful for noisy images and supports user-defined markers for faster assessment. | Demands substantial computational resources and memory. |
| Watershed Segmentation (Jin et al., 2018) | A specific form of region segmentation that uses image topology to create boundary-based segments. | Yields well-defined segment borders. | Determining ridge gradients for segmentation is complex. |

### 2.3.4. Techniques in Machine Vision and Learning Models

Machine Vision refers to the ability of a computer to interpret, analyze, and make decisions based on visual data. Utilizing cameras, digital signals, and analytical algorithms, Machine Vision systems can automate tasks that require visual inspection, such as quality control in manufacturing, facial recognition, medical imaging, and more (Mahadevkar et al., 2022).

1. Processes in Machine Vision
    - Image Acquisition: This is the initial stage where an image is captured using cameras or other imaging devices.
    - Preprocessing: This stage involves cleaning and transforming raw image data into a format suitable for further analysis.

- Image Analysis: Algorithms are employed to scrutinize the image, identifying patterns, shapes, or specific objects within it.
- Interpretation and Decision Making: Based on the analysis, the system interprets the information and makes decisions or actions accordingly.

Machine Learning falls within the domain of Artificial Intelligence, enabling systems to learn from data rather than relying solely on explicit programming. It encompasses a range of algorithms and statistical models empowering computers to execute tasks without explicit instructions.

2. Types of Machine Learning
   - Supervised Learning: The model learns from labeled training data and makes predictions or decisions based on input data.
   - Unsupervised Learning: Without labeled responses to guide the learning process, this model identifies patterns and structures from the data itself.
   - Reinforcement Learning: This approach entails learning through trial and error, wherein an agent makes decisions and receives feedback in the form of rewards or penalties based on its actions.

#### *2.3.4.1. Different Models and Algorithms*

1. Linear Regression: Linear regression is a fundamental statistical technique used for predicting numerical outcomes based on the relationship between one or more independent variables (predictors) and a dependent variable (response). The primary goal of linear regression is to find the best-fitting line that minimizes the overall distance between the data points and the regression line (Schneider et al., 2010). The mathematical representation of a linear regression model is given by Equation 2.1:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn + \varepsilon \quad (2.1)$$

Where, y is the dependent variable (response), β0 is the intercept or bias term, β1, β2,…,βn are the coefficients of the independent variables x1,x2,…,xn, ε represents the error term, which accounts for the difference between the predicted and actual values.

The coefficients $\beta 1, \beta 2, \ldots, \beta n$ are estimated using methods like Ordinary Least Squares (OLS) or Gradient Descent, which minimize the sum of squared residuals.

Linear regression is widely used in various fields, such as economics, finance, social sciences, and machine learning. It serves as a building block for more complex regression models and is useful for both understanding the relationships between variables and making predictions.

2. Logistic Regression: Logistic regression is a classification algorithm used to predict binary outcomes (0 or 1) based on one or more independent variables. It is an extension of linear regression, but instead of predicting continuous values, it predicts the probability that an instance belongs to a particular class (X. Zou et al., 2019). The mathematical representation of logistic regression is given by the logistic function (sigmoid) shown in Equation 2.2:

$$P(y = 1) = 1/1 + e^{-z} \qquad (2.2)$$

Where, $P(y = 1)$ is the probability that the outcome is 1 (belongs to the positive class), $z = \beta 0 + \beta 1 x1 + \beta 2 x2 + \ldots + \beta n xn$ is the linear combination of coefficients and independent variables, $e$ is the base of the natural logarithm.

The logistic function maps the output of the linear regression model to a value between 0 and 1, which represents the probability of the positive class. Based on a predefined threshold (usually 0.5), the instances are classified into either the positive or negative class.

Logistic regression is commonly used in fields like healthcare (disease diagnosis), marketing (customer churn prediction), and natural language processing (sentiment analysis). It is an essential tool in the toolkit of machine learning algorithms for classification tasks.

3. Decision Trees: Decision trees are non-parametric supervised learning models used for both classification and regression tasks. These models recursively split the data based on the values of input features, creating a tree-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label or a numeric value (Rokach & Maimon, n.d.).

   The decision-making process in a decision tree starts from the root node and moves down through the tree along edges, based on the feature values. The splitting criteria aim to maximize the information gain or minimize impurity, depending on the type of task (classification or regression).
   Decision trees are favored for their interpretability, as they provide transparent decision-making processes that can be visualized and understood easily. However, they tend to overfit the data if not pruned properly, leading to poor generalization on unseen data.
4. Random Forest: Random Forest is an ensemble learning method based on the combination of multiple decision trees (Han et al., 2018). It aims to improve the accuracy and robustness of decision tree models by introducing randomness in two main ways:

   *a) Bootstrapped Sampling:* For each tree in the forest, a random subset of the original data is selected through bootstrapping (sampling with replacement). This results in each tree being trained on slightly different data, introducing diversity.

   *b) Feature Randomness:* At each split in a tree, a random subset of features is considered for the splitting criteria. This prevents a single dominant feature from driving all decisions, further increasing diversity.

   Random Forest aggregates the predictions of individual trees, either through majority voting (for classification) or averaging (for regression), to arrive at the final prediction. The ensemble nature of Random Forest helps reduce overfitting and improves generalization.

   Random Forests are widely used in various domains, such as image classification, customer churn prediction, and anomaly detection. They are

considered one of the most powerful and versatile machine learning algorithms.

5. Support Vector Machines (SVM): Support Vector Machines (SVM) are powerful supervised learning algorithms used for both classification and regression tasks. SVM focuses on finding the hyperplane that best separates data points of different classes in a high-dimensional space. In the case of linearly separable data, SVM finds the optimal hyperplane that maximizes the margin (distance) between the two classes (Sari et al., 2019).

   When the data is not linearly separable, SVM uses a technique called the kernel trick to transform the data into a higher-dimensional space, where it becomes linearly separable. Common kernel functions include polynomial kernels, radial basis function (RBF) kernels, and sigmoid kernels.

   The primary advantage of SVM lies in its ability to handle high-dimensional data effectively and perform well even on small datasets. Additionally, SVM can handle both linear and nonlinear decision boundaries. However, SVM's main limitation is its sensitivity to the choice of hyperparameters and the computational cost, particularly when dealing with large datasets. SVM finds applications in various domains, such as image recognition, text classification, and bioinformatics.

6. Neural Networks: Neural Networks, inspired by the human brain's neural structure, are a class of deep learning models used for various tasks, including image and speech recognition, natural language processing, and reinforcement learning. They are composed of interconnected artificial neurons organized in layers (Macukow, 2016).

   The basic building block of a neural network is a neuron or node, which receives inputs, applies a weighted sum, and passes the result through an activation function to produce an output. Neural networks consist of an input layer, one or more hidden layers, and an output layer.

   During training, neural networks adjust their weights and biases through a process called backpropagation. This process uses optimization techniques like stochastic gradient descent to minimize the difference between the predicted outputs and the actual labels.

Deep neural networks, known as deep learning models, have demonstrated impressive capabilities in handling large-scale, complex data and have achieved state-of-the-art performance in various domains, including computer vision, natural language processing, and game-playing.

7. k-Means Clustering: k-Means Clustering is an unsupervised machine learning algorithm used for partitioning data into k clusters, where each data point belongs to the cluster with the nearest mean. The algorithm aims to minimize the variance within each cluster while maximizing the variance between clusters (Said & Jambek, 2021). The k-means algorithm works as follows:
    1. Randomly initialize k centroids.
    2. Assign each data point to the nearest centroid based on the Euclidean distance.
    3. Recalculate the centroids as the mean of all points assigned to each cluster.
    4. Repeat steps 2 and 3 until the centroids converge or a predefined number of iterations is reached.

    The final result of the k-means algorithm is a set of k clusters, each represented by its centroid. These centroids act as prototypes or representatives for their respective clusters. The k-means algorithm converges when the centroids stabilize, and the data points are relatively well-clustered around their assigned centroids.

    One of the primary challenges in k-means clustering is determining the appropriate value of k, which represents the number of clusters. Picking an inappropriate value of k can lead to suboptimal clustering results. There are various methods, such as the elbow method and silhouette score, to help in selecting the optimal value of k based on the data.

    Although k-means is straightforward and computationally efficient, it has some limitations. Firstly, it assumes that clusters are spherical and have equal variance, which may not be suitable for complex-shaped or unevenly distributed data. Secondly, k-means can be sensitive to the initial placement of centroids, which can lead to different clustering results.

    To overcome some of these limitations, there are variations of k-means, such as k-medoids (PAM: Partitioning Around Medoids), which use the

medoid (most central point) instead of the centroid to represent the cluster. This makes k-medoids more robust to outliers and data with non-spherical shapes. K-means clustering finds applications in various fields, including image segmentation, customer segmentation, anomaly detection, and market research.

8. Reinforcement Learning Models like Q-learning: Reinforcement Learning (RL) is a type of machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent's goal is to maximize a cumulative reward over time, making sequential decisions in the presence of uncertainty.

   One of the fundamental algorithms used in RL is Q-learning, which is suitable for problems with discrete states and actions. The Q-learning algorithm aims to learn an action-value function (Q-function) that estimates the expected reward an agent will receive for taking a particular action in a given state (Jang et al., 2019).

   The Q-function is updated iteratively based on the agent's experiences during the learning process. At each time step, the agent observes the current state, takes an action based on an exploration-exploitation strategy (e.g., epsilon-greedy), receives a reward, and transitions to a new state. The Q-function is then updated using the Bellman equation, which recursively incorporates the reward and the maximum Q-value of the next state.

   Over time, as the agent explores the environment and learns from its experiences, the Q-function converges to the optimal action-value function, which allows the agent to make informed decisions that lead to the maximum expected cumulative reward.

   One of the hurdles in Q-learning involves managing high-dimensional state spaces, a factor that can significantly increase the computational complexity and slow down the learning process. To address this, deep reinforcement learning methods combine Q-learning with deep neural networks, creating Deep Q-Networks (DQNs). DQNs have demonstrated remarkable success in solving complex RL problems, such as playing video games and controlling robotic systems.

Q-learning and its extensions are applied in various real-world scenarios, including autonomous vehicles, robotic control, recommendation systems, and game playing.

### 2.3.5. Deep Learning and Its Models

Deep learning is a subfield of machine learning that focuses on building and training artificial neural networks with multiple layers (deep architectures). These networks are inspired by the structure and function of the human brain and are capable of learning complex patterns and representations from data (Sarker, 2021). Deep learning has gained immense popularity in recent years due to its remarkable performance in various tasks such as image recognition, natural language processing, and game playing. Key characteristics of deep learning include.

1. **Neural Networks:** At the core of deep learning are artificial neural networks, which consist of interconnected nodes (neurons) organized in layers. Each neuron receives input data, applies a weighted sum, and passes the result through an activation function to produce an output. The neurons in the subsequent layers build upon the outputs of the previous layers, allowing for the learning of complex features and abstractions.
2. **Deep Architectures:** Deep learning models have multiple hidden layers between the input and output layers. These deep architectures allow the network to learn hierarchical representations of data, capturing intricate patterns and features at different levels of abstraction. The ability to learn from multiple layers enables deep learning models to handle increasingly complex and challenging tasks.
3. **Feature Learning:** One of the significant advantages of deep learning is its ability to automatically learn relevant features from the raw input data. Unlike traditional machine learning approaches where feature engineering is critical, deep learning alleviates the need for manual feature extraction, making it more scalable and flexible. Deep learning models can effectively learn and discover relevant features during the training process.
4. **End-to-End Learning:** Deep learning models are often trained end-to-end, meaning they directly learn the mapping from input to output without

explicitly decomposing the problem into intermediate steps. This approach simplifies the learning process and reduces the risk of error propagation across different stages. End-to-end learning has proven particularly useful in computer vision and natural language processing tasks.

#### *2.3.5.1. Various Deep Learning Models*

1. **Convolutional Neural Networks (CNNs):** CNNs are widely used for image and video recognition tasks. They utilize convolutional layers to apply filters to input images, capturing local patterns and features such as edges, corners, and textures. The pooling layers downsample the spatial dimensions, reducing the computational complexity while retaining essential information. CNNs have shown exceptional performance in image classification, object detection, and image segmentation tasks (Alzubaidi et al., 2021).

   In CNN, the convolutional layer applies a set of learnable filters (kernels) to the input image to create feature maps. Given an input image *X*, the convolution operation at a specific position (*i*,*j*) in the feature map *F* can be defined as in Equation 2.3:

$$Fij = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} Wmn \cdot X(i+m),(j+n) + b \qquad (2.3)$$

   Where, *Wmn* is the weight of the kernel at position (*m*,*n*), *X*(*i*+*m*),(*j*+*n*) represents the pixel value of the input image at position (*i*+*m*,*j*+*n*), *b* is the bias term.

   The pooling layer then performs downsampling, typically through max pooling, to reduce the spatial dimensions of the feature maps. The max-pooling operation can be defined as Equation 2.4:

$$Pij = max(X(2i),(2j),X(2i),(2j+1),X(2i+1),(2j),X(2i+1),(2j+1)) \qquad (2.4)$$

   Where, *Pij* is the pooled value at position (*i*,*j*) in the pooled feature map.

2. **Residual Networks (ResNets)** represent a significant breakthrough in the field of deep learning and have profoundly impacted the design and performance of neural networks. Introduced by (He et al., 2015a), ResNets addressed the critical issue of vanishing gradients, which often hindered the training of deep neural networks. The core innovation of ResNets is the introduction of residual learning, where shortcut connections, or skip connections, are employed to bypass one or more layers.

   In traditional neural networks, as the network depth increases, the gradients tend to diminish, making it difficult to train deep models effectively. This often results in degradation problems, where adding more layers leads to higher training and test errors. ResNets tackle this problem by allowing gradients to flow directly through these shortcut connections, ensuring that the essential features are preserved and propagated through the network.

   The architecture of a typical residual block includes a series of convolutional layers followed by an addition operation that merges the input and the output of these layers. This approach enables the network to learn residual functions with reference to the layer inputs, rather than learning unreferenced functions. In essence, ResNets learn the difference between the input and the output, rather than the entire transformation, which simplifies the learning process.

   To illustrate the working mechanism, consider a simple residual block with two convolutional layers. The input x to the block undergoes convolution, batch normalization, and a ReLU activation function to produce an intermediate output. This intermediate output is then passed through another set of convolutions, batch normalization, and activation layers. Instead of directly using this final output, ResNets add the original input x to the final output, creating the output y. Mathematically, this can be represented as Equation 2.5:

$$y = F(x, \{Wi\}) + x \tag{2.5}$$

where $F(x, \{Wi\})$ represents the function learned by the convolutional layers with weights $\{Wi\}$. This identity mapping ensures that even if the learned function $F(x, \{Wi\})$ approaches zero, the information from *x* is still propagated through the network. This mechanism significantly eases the training of deep networks, allowing them to achieve better performance and convergence.

ResNets have been proven to significantly improve the performance of deep neural networks, allowing for the successful training of models with hundreds or even thousands of layers. This advancement has led to state-of-the-art results in various computer vision tasks, including image classification, object detection, and segmentation. The ability to train ultra-deep networks has also spurred further research and innovations, such as DenseNets and other architectures that build upon the principles of residual learning.

3. **Recurrent Neural Networks (RNNs):** RNNs are designed to handle sequential data, such as time series, text, and speech. RNNs have feedback connections that allow them to retain information from previous time steps, enabling them to model temporal dependencies effectively. However, traditional RNNs can suffer from the vanishing gradient problem, limiting their ability to capture long-term dependencies. To address this, variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were introduced, which have improved memory retention and gradient flow (Sherstinsky, 2020).

   The hidden state *ht* at time step *t* in an RNN is updated using the input *xt* and the previous hidden state *ht*−1 as Equation 2.6:

$$ht = Activation(Wih \cdot xt + Whh \cdot ht - 1 + bh) \qquad (2.6)$$

   where, *Wih* is the weight matrix between the input and hidden layer, *Whh* is the weight matrix between the hidden layer and itself, *bh* is the bias term for the hidden layer.

   However, traditional RNNs can suffer from the vanishing gradient problem, limiting their ability to capture long-term dependencies. To

address this, variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) were introduced. In LSTM, the update equations are given as 2.7, 2.8, 2.9, 2.10, 2.11 and 2.12:

$$ft = \sigma(Wif \cdot xt + Whf \cdot ht - 1 + bf) \quad (2.7)$$

$$it = \sigma(Wii \cdot xt + Whi \cdot ht - 1 + bi) \quad (2.8)$$

$$ot = \sigma(Wio \cdot xt + Who \cdot ht - 1 + bo) \quad (2.9)$$

$$\tilde{c}t = tanh(Wic \cdot xt + Whc \cdot ht - 1 + bc) \quad (2.10)$$

$$ct = ft \odot ct - 1 + it \odot \tilde{c}t \quad (2.11)$$

$$ht = ot \odot tanh(ct) \quad (2.12)$$

where, $\sigma$ is the sigmoid activation function, $\odot$ represents element-wise multiplication.

4. **Generative Adversarial Networks:** GANs comprise two neural networks: a generator (G) and a discriminator (D), engaged in a competitive game against each other. The generator tries to generate realistic data samples *G*(*z*) given a random noise vector *z*, while the discriminator aims to distinguish between real data samples and generated samples. Through this adversarial training, GANs can create high-quality synthetic data, leading to impressive results in image synthesis, style transfer, and data augmentation. GANs have also been employed in generating realistic deepfake videos and art generation (Fan & Hu, 2019). The objective function for GANs is defined as Equation 2.13:

$$minGmaxDV(D,G) = Ex \sim pdata(x)[logD(x)] + Ez \sim pz(z)[log(1 - D(G(z)))] \quad (2.13)$$

Where, *p*data(*x*) is the distribution of real data samples, *pz*(*z*) is the distribution of the noise vector *z*.

5. **Transformer Networks:** Transformers have revolutionized natural language processing tasks. They use an attention mechanism to weigh the importance of different words in a sentence, enabling them to model long-range dependencies effectively. Transformers have shown state-of-the-art

performance in machine translation, language modeling, question-answering systems, and text generation (Lin et al., 2022). Pretrained transformer models like BERT (Bidirectional Encoder Representations from Transformers) have become popular for transfer learning in various NLP tasks.

The attention mechanism in transformers calculates the attention weights between each word in the input sequence *X* and all other words in the sequence, generating the attention matrix *A* as in Equation 2.14:

$$Aij = exp(score(Qi, Kj)) \,/\, \sum_{k=1}^{N} exp(score(Qi, Kk)) \quad (2.14)$$

Where, *Qi*, *Kj* are the query and key representations of word *i* and *j*, *N* is the total number of words in the sequence, score (*Qi*, *Kj*) = $Qi \cdot K_j^T$.

Attention weights *A* determines the importance of each word in the sequence concerning all other words. The transformer then calculates the weighted sum of the values *Vj* to obtain the output representation for each word *i is* given in Equation 2.15:

$$Output_i = \sum_{j=1}^{N} Aij \cdot Vj \quad (2.15)$$

Transformers have shown state-of-the-art performance in machine translation, language modeling, question-answering systems, and text generation. Pretrained transformer models like BERT (Bidirectional Encoder Representations from Transformers) have become popular for transfer learning in various NLP tasks.

6. **Autoencoders:** Autoencoders are unsupervised learning models used for feature learning and data compression. They consist of an encoder that maps input data to a latent space representation and a decoder that reconstructs the input from the latent space (Berahmand et al., 2024). Autoencoders are useful for dimensionality reduction, denoising, and anomaly detection. Variational Autoencoders (VAEs) add a probabilistic

component, enabling them to generate new data samples from the learned distribution.

7. **Reinforcement Learning Models:** Deep reinforcement learning combines deep neural networks with reinforcement learning algorithms. These models have been successful in achieving superhuman performance in games like AlphaGo and Dota 2. They are also applied in robotics, finance, and control systems. Deep Q-Networks (DQNs) and Proximal Policy Optimization (PPO) are popular deep reinforcement learning algorithms (Sivamayil et al., 2023).
8. **YOLO:** Introduced by (Redmon et al., 2015), YOLO brought a paradigm shift in object detection by emphasizing speed and accuracy. Unlike traditional methods that apply models to multiple regions of interest, YOLO processes the entire image in a single pass, significantly reducing computation time.

    - **YOLOv1: The Beginning**

      YOLOv1 redefined object detection by proposing a unified architecture. It divides the image into an SxS grid, where each grid cell predicts B bounding boxes and confidence scores. Each bounding box consists of five predictions: x, y, width, height, and confidence. The confidence score reflects the accuracy of the bounding box and whether it contains an object. This version achieved impressive speeds but struggled with small objects and localization errors.

      Key Innovations:

      Unified detection architecture.

      Real-time processing with 45 frames per second (fps) on PASCAL VOC.

    - **YOLOv2: The Refinement**

      YOLOv2, also known as YOLO9000, introduced several improvements:

      Batch Normalization: This normalization technique accelerated convergence and improved accuracy.

      High-Resolution Classifier: Training the classifier at higher resolution enhanced the detector's performance.

Anchor Boxes: Borrowed from Faster R-CNN, anchor boxes improved the detection of objects with varied dimensions.

Multi-Scale Training: Training the network at different scales made it more robust to scale variations.

YOLOv2 achieved state-of-the-art performance on standard benchmarks while maintaining high speeds.

- **YOLOv3: The Leap Forward**

  YOLOv3 further improved upon its predecessors by:

  Feature Pyramid Network (FPN): Enabling detection at three different scales, enhancing the detection of both large and small objects.

  Darknet-53 Backbone: A more powerful feature extractor, replacing Darknet-19, with 53 convolutional layers.

  Bounding Box Predictions: Using logistic regression for predicting objectness scores, improving localization accuracy.

  YOLOv3 balanced speed and accuracy, achieving 30 fps on a Titan X GPU while maintaining high detection performance.

- **YOLOv4: Performance Optimization**

  YOLOv4 focused on optimizing speed and accuracy for real-time object detection:

  Bag of Freebies (BoF): Techniques like Mosaic data augmentation, Self-Adversarial Training (SAT), and Class Label Smoothing improved training without additional inference costs.

  Bag of Specials (BoS): Enhancements like Mish activation, Cross-stage Partial connections (CSP), and Spatial Pyramid Pooling (SPP) modules increased network capability.

  CSPDarknet53 Backbone: A further refined backbone for better feature extraction.

  YOLOv4 achieved significant improvements in AP (Average Precision) and fps, becoming a popular choice for real-time applications.

- **YOLOv5: Towards User-Friendly Implementation**

  Developed by Ultralytics, YOLOv5 introduced ease of use and deployment as key priorities:

PyTorch Implementation: Making it accessible and easier to integrate into existing workflows.

AutoAnchor and Hyperparameter Evolution: Automatically optimizing anchor boxes and hyperparameters during training.

Lightweight Models: Variants like YOLOv5s (small) for resource-constrained environments.

YOLOv5 continued the trend of combining accuracy and efficiency, with extensive community support and rapid iteration.

- **YOLOv6: Enhanced Performance**

  YOLOv6 focused on achieving superior performance while maintaining efficiency:

  Advanced Backbone Network: Improved feature extraction capabilities.

  Efficient Training Techniques: Incorporating new data augmentation and loss functions.

  Scaled Models: Providing different model sizes to cater to various application needs, from edge devices to high-performance servers.

  YOLOv6 aimed to bridge the gap between lightweight models and high-accuracy detection.

- **YOLOv7: Cutting-Edge Innovations**

  YOLOv7 introduced state-of-the-art innovations for real-time detection:

  Extended Feature Extractor: Enhancing the network's ability to detect small and complex objects.

  Optimized Inference: Further reducing latency and improving throughput.

  Improved Post-Processing: Refinements in non-maximum suppression (NMS) and other post-processing steps to increase detection accuracy.

  YOLOv7 set new benchmarks for speed and accuracy in the object detection landscape.

  The selection of YOLOv7 for our analysis is justified by its cutting-edge attributes, including Extended Efficient Layer Aggregation (E-

ELAN), Model Scaling Techniques, Re-parameterization Strategy, and the Auxiliary Head Coarse-to-Fine method. A crucial aspect of the YOLO architecture is the efficiency of its convolutional layers within the backbone, facilitating swift processing. The creators of YOLOv7 have refined this aspect by building on previous developments, considering the gradient path length necessary for back-propagation and the memory allocation for storing layers. Notably, the efficiency of network training is enhanced by minimizing the gradient size required. E-ELAN, an advanced iteration of the ELAN computational block, was selected for its layer aggregation capability, a key feature depicted in Figure 2.6.

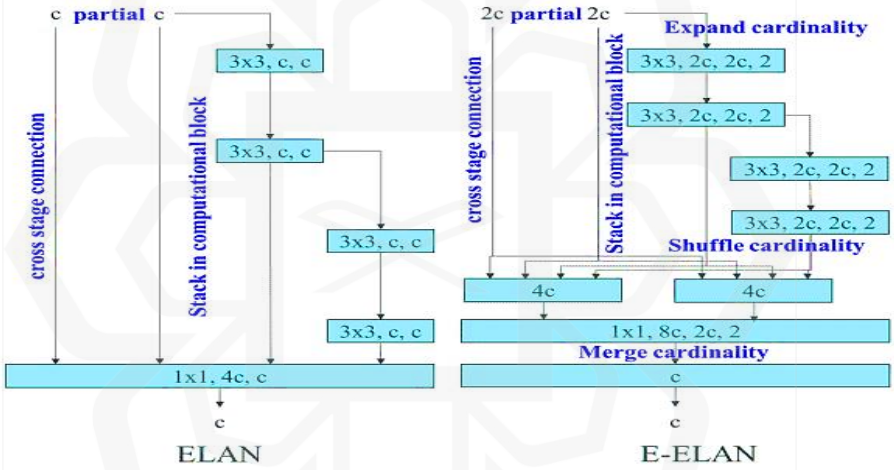



Figure 2.6. Advancements in Layer Aggregation Techniques within the YOLOv7 Framework (C.-Y. Wang et al., 2022)

In object detection algorithms, standard metrics such as network depth, breadth, and resolution are often considered during the training phase. The YOLOv7 model introduces an innovative approach by scaling both the depth and width of the network concurrently as it combines layers, a process illustrated in Figure 2.7. This methodology maintains an ideal architectural balance, ensuring model efficacy irrespective of scale adjustments, either increasing or decreasing in size.

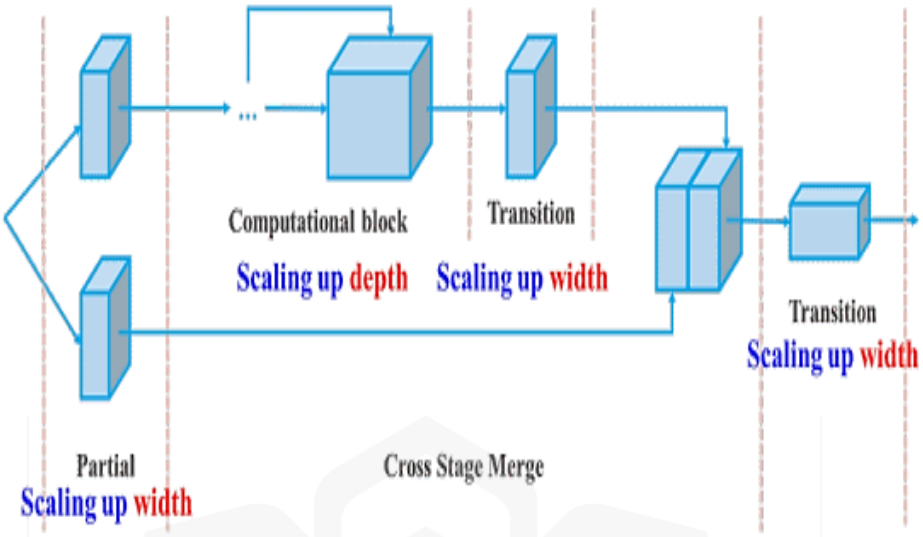



Figure 2.7. Scaling in YOLOv7 (C.-Y. Wang et al., 2022)

Re-parameterization in this context involves refining a model's weight distribution to bolster its capability in recognizing general patterns, enhancing overall robustness. Recent advancements have shifted towards module-level re-parameterization, where different components within the network adopt unique re-parameterization strategies tailored to their functions. YOLOv7 strategically selects network modules for re-parameterization by analyzing the pathways of gradient flow, aiming to optimize the learning process.

While the YOLO framework traditionally relies on its final 'head' layer for making predictions, the introduction of an auxiliary head situated closer to the network's core presents distinct advantages. This setup allows for parallel observations of the prediction and detection heads during training. Given its fewer intervening layers, the auxiliary head benefits from more direct supervision, leading to a more effective learning process. The YOLOv7 team employed a hierarchical supervision strategy, where feedback to the auxiliary head varies in detail, adopting a coarse-to-fine methodology. This

approach significantly boosts the model's training efficiency, a concept depicted in Figure 2.8.

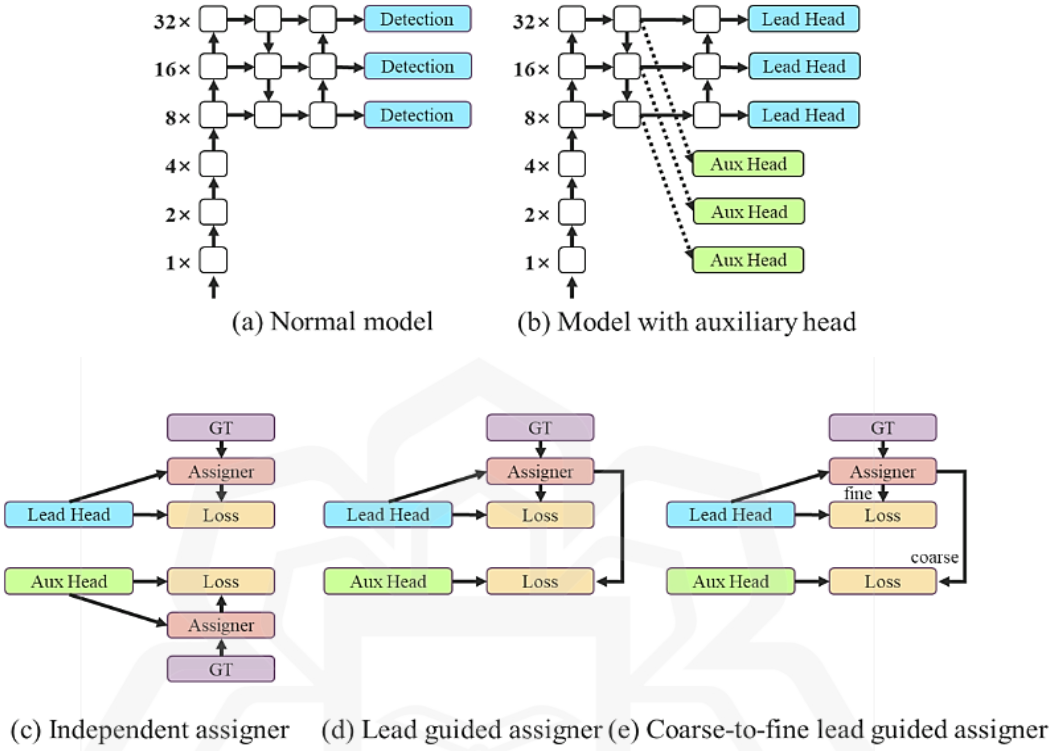



Figure 2.8. Hierarchical Supervision Strategy for the Auxiliary Head in YOLOv7 (C.-Y. Wang et al., 2022)

- **YOLOv8: The Current Frontier**

  YOLOv8 continued to build on the strengths of its predecessors:

  Hybrid Architecture: Combining elements from previous YOLO versions and integrating new neural network components.

  Adaptive Learning: Implementing advanced learning algorithms to enhance model adaptability and generalization.

  Comprehensive Deployment Options: Facilitating deployment across diverse environments, including cloud, edge, and mobile devices.

  Significant advancement in the YOLO lineage, bridging the gap between object detection, tracking, instance segmentation, image classification and pose estimation tasks. The performance comparison

of YOLOv8 as compared to previous versions is represented in Figure 2.9.

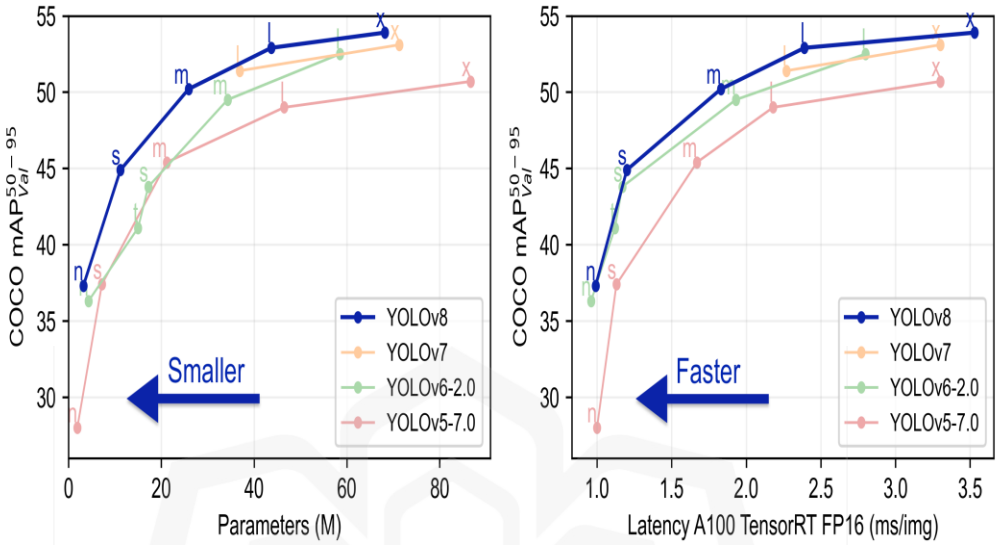



Figure 2.9. YOLOv8 performance compared to previous versions.

While traditional YOLO models were confined to object detection, YOLOv8x extends to instance segmentation, a task that involves not only recognizing the objects in an image but also precisely outlining each one (Jocher et al., 2023). The model predicts a pixel-wise mask for each detected object, offering a more granular understanding of the scene. In the context of this study, the chosen architecture is YOLOv8x-seg. The decision to employ this specific customization stems from a thorough analysis of its performance metrics relative to other segmentation architectures, as depicted in Table 2.4. This customized version showcases unique characteristics that make it well-suited for the task at hand, reflecting a balance of efficiency and effectiveness that sets it apart from alternative models. Its specialized design allows for more precise targeting of the research objectives, leveraging the strengths of YOLOv8 and optimizing them for the particular demands of the segmentation task involved in this study.

Table 2.4. Comparison of YOLOv8 Segmentation Architectures for Object Detection and Segmentation Performance

| Model Variation | Resolution | Box mAP (50-95) | Mask mAP (50-95) | Inference Time on CPU (ms) | Inference Time on A100 GPU (ms) | Model Parameters (Millions) | Computational Complexity (Billions of FLOPs) |
|---|---|---|---|---|---|---|---|
| YOLOv8n seg | 640 pixels | 36.7 | 30.5 | 96.1 | 1.21 | 3.4 | 12.6 |
| YOLOv8s-seg | 640 pixels | 44.6 | 36.8 | 155.7 | 1.47 | 11.8 | 42.6 |
| YOLOv8m-seg | 640 pixels | 49.9 | 40.8 | 317.0 | 2.18 | 27.3 | 110.2 |
| YOLOv8l-seg | 640 pixels | 52.3 | 42.6 | 572.4 | 2.79 | 46.0 | 220.5 |
| YOLOv8x-seg | 640 pixels | 53.4 | 43.4 | 712.1 | 4.02 | 71.8 | 344.1 |

- **YOLOv9: The Latest Advancements**

  YOLOv9, the latest iteration in the YOLO series, introduces groundbreaking advancements:

  Programmable Gradient Information (PGI): Ensuring the preservation of essential data across deep network layers, facilitating accurate model updates and improving overall detection performance.

  Generalized Efficient Layer Aggregation Network (GELAN): Enabling superior parameter utilization and computational efficiency, allowing for flexible integration of various computational blocks.

  Information Bottleneck Principle: Addressing the challenges posed by information loss in deep neural networks, ensuring the retention of crucial information throughout the detection process.

Reversible Functions: Mitigating the risk of information degradation, especially in deeper layers, by allowing the network to retain a complete information flow.

Impact on Lightweight Models: Ensuring that even with a streamlined model, the essential information required for accurate object detection is retained and effectively utilized.

YOLOv9 sets a new standard in real-time object detection, pushing the boundaries of both speed and accuracy.

### 2.3.6. Object Detection Models Utilizing Machine Vision and Learning Techniques

In the domain of computer vision, the methodology of object detection serves to ascertain the existence of objects within images or video content. This technique predominantly leverages machine learning or deep learning frameworks to yield pertinent outcomes. The inherent capability of humans to swiftly discern and categorize various objects within visual media is a faculty that object detection endeavors to emulate within computational systems. The discipline of computer vision, specifically the aspect concerning object identification, has witnessed significant advancements in recent times. At its core, object detection entails the determination of object positions within an image, a task known as object localization. Concurrently, it aims to classify the objects detected, a process referred to as object classification. As evidenced in Figure 2.10, there has been a notable increase in scholarly contributions in this area, incorporating machine vision and machine learning techniques.

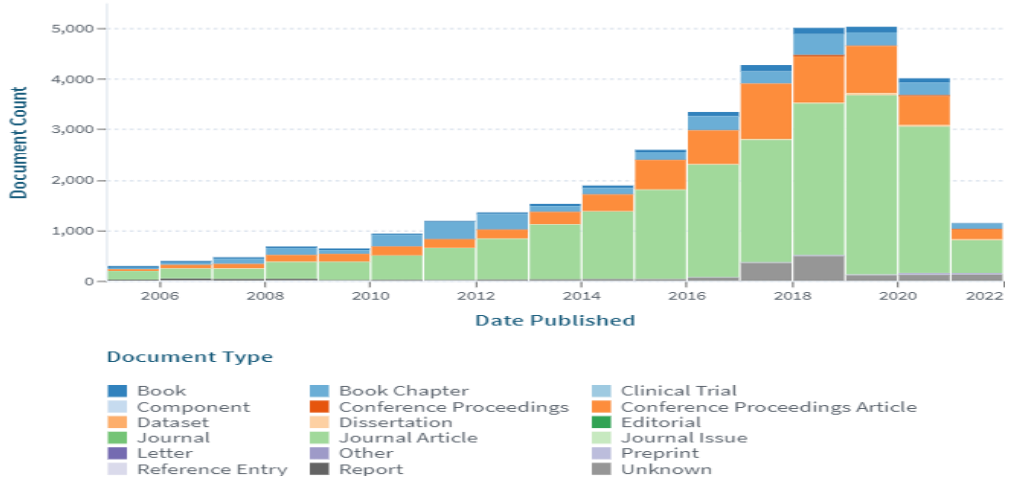



Figure 2.10. Evolution of Research in Machine Learning-Based Object Detection *(Accessed from Lens.org on 28/03/2022)*

The predominant methodologies for object identification incorporate techniques such as sliding window approaches and super-pixel segmentation, exemplified by Multiscale Combinatorial Grouping (Arbelaez et al., 2014), Constrained Parametric Min-Cuts (CPMC), Selective Search, EdgeBoxes (Zitnick & Dollár, 2014), and Objects in Windows. The R-CNN framework (Girshick et al., 2014) stands out for its effectiveness in delineating areas as either objects or background within the object detection process. Innovations by Viola and Jones (Viola & Jones, n.d.) have refined sliding window techniques through the application of Haar features. Additionally, the integration of Histogram of Oriented Gradients (HOG) features (Dalal & Triggs, n.d.) with linear Support Vector Machines (SVMs), coupled with the employment of Deformable Part Models (DPM), has facilitated the construction of deformable models. The OverFeat method (Sermanet et al., 2013) utilizes sliding windows on a convolutional feature map, interfaced with a fully connected layer, to enhance detection and classification efficacy. In the realm of Spatial Pyramid Pooling (SPP)-based detection, the aggregation of features from designated regions on the convolutional feature map, followed by their introduction to a fully connected layer, aids in the classification process (He et al., 2015b).

### 2.3.7. Methods for Identifying, Categorizing, and Analyzing Pavement Cracks

#### *2.3.7.1. Traditional Approaches*

Before the adoption of machine learning technologies in the field of pavement crack analysis, traditional approaches were the norm. These earlier methods, which predate the era of automation, were characterized by their significant demand for manual labor and extensive time investment. Figure 2.11 illustrates the contributions of numerous researchers to the area of road crack detection using these conventional techniques. The primary methods of crack detection involved manual inspection or sensor-based techniques, with a focus primarily on crack sealing and filling operations (CHUN et al., 2015). The standard protocol involved obtaining construction maintenance records for the selection of design and repair strategies based on road age, followed by manual surveys to detect cracks. These surveys aimed to document the extent and severity of pavement cracks.

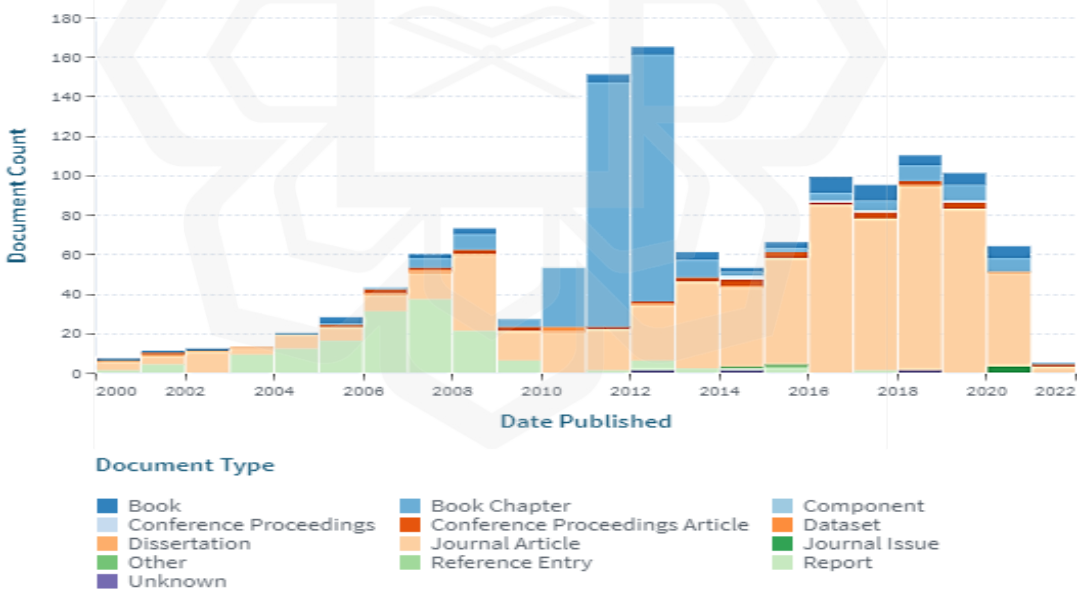



Figure 2.11. Illustration of Traditional Pavement Crack Detection *(Accessed from Lens.org on 16/04/2022)*

As reported by (Jo & Ryu, 2015), damage to concrete pavement structures can arise from both dynamic and static pressures. The application of a four-point bending

assessment on standard concrete pavement specimens has been utilized to gauge the endurance and responsiveness of integrated sensors, aiding in the identification and monitoring of crack development. Various techniques have been applied to analyze ultrasonic signals, assessing changes within structures by examining the signals' characteristics. Studies have demonstrated that ultrasonic sensors can achieve 100% accuracy in crack detection before they become visible to the naked eye, even if the crack is not directly in the path of the ultrasonic wave. The results confirm the feasibility of early crack detection using this methodology. Further research has developed detection systems for identifying and monitoring internal crack propagation using discrete strain sensors embedded within the pavement. Theoretical approaches, based on linear elastic fracture mechanics, have been derived to locate bottom-up cracks and track their progression using two or more discrete in-pavement strain sensors. Experimental findings have shown that this crack detection method, utilizing two discrete strain sensors, can accurately identify bottom-up cracks with an average precision of 82.4%.

Cracking in concrete pavements is a critical concern for their performance, particularly the emergence of internal bottom-up cracks, which can lead to water infiltration within the pavement structure and subsequent degradation. Prompt detection of such cracks in concrete pavements facilitates timely maintenance, thereby enhancing the integrity and longevity of the infrastructure. Various researchers have explored crack detection methodologies, employing diverse techniques. For instance, employing a dynamic thresholding approach, a study (A. Zhang et al., 2017) succeeded in identifying potential cracks as dark pixels within images. This research utilized entropy analysis to segment thresholded images into distinct, non-overlapping sections, creating an entropy block matrix to pinpoint image segments containing potential crack pixels. Moreover, another study (Zalama et al., 2014) advocated for a dual-threshold technique, applying a sophisticated Otsu edge segmentation algorithm to remove road markings from images, followed by an enhanced, flexible iterative threshold segmentation algorithm to isolate the remaining image segments. Subsequent morphological denoising processes allowed for the delineation of the crack outlines. In another innovation, a multiscale optimal edge segmentation algorithm was proposed (Akarsu et al., 2016) for segmenting asphalt cracks based on

their thickness distribution, achieving superior segmentation results compared to conventional global thresholding methods.

Edge detection strategies also play a pivotal role in the identification of cracks within pavement structures. (Amhaz et al., 2016) introduced an advanced Smart Edge Detection method for the delineation of roadway boundaries. This method employs wavelet transformation to enhance the visibility of obscured edges, coupled with a sophisticated adaptive thresholding algorithm optimized through genetic algorithms. Furthermore, research conducted by (H. Li et al., 2019) focused on a crack detection approach that integrates bi-dimensional empirical mode decomposition with Sobel edge detection. The bi-dimensional empirical mode decomposition serves as an advancement of the standard empirical mode decomposition, effectively reducing noise in the signal without the reliance on complex convolutional techniques. While this edge detection algorithm is capable of mapping the distribution and outlines of crack deformities, it is less effective in accurately capturing the internal details of the cracks.

Regular inspections and maintenance are imperative for effective asphalt preservation, as cracks not only impair the roadway's appearance and smoothness but also diminish its lifespan. To efficiently detect cracks, a novel detection method was developed based on a grid network (Cao et al., 2018), utilizing a segmentation network alongside corner-based detection to facilitate crack extraction. This approach employed the quantitative characteristics of cracks to establish thresholds for their length and width, achieving error margins of 4.23% and 6.98%, respectively, compared to actual measurements. Furthermore, a programmed system for conducting asphalt crack analyses was suggested (Fan et al., 2018a), leveraging historical crack data as a benchmark. Initially, a multiscale constraint approach was employed to align images of recorded and inspected cracks, incorporating GPS-based coarse localization, image-level alignment, and metric constraints. Verified crack pixels were subsequently mapped onto the designated crack images, acting as initial points for further crack investigation. Improvements in identifying cracks were realized through the adoption of the Region Growing Method, which facilitated the discovery of emerging cracks. This method was corroborated using actual asphalt images gathered over a period, achieving an F-measure of 88.9% for tracking crack growth.

### *2.3.7.2. Techniques Based on Machine Vision and Machine Learning*

The study of detecting pavement cracks has greatly benefited from the integration of machine vision and machine learning techniques. This field has experienced a remarkable upswing in research interest, propelled by the rapid evolution of recognition technologies and the increasing automation capabilities. The last decade, in particular, has seen a notable escalation in academic endeavors within this domain, a development that is comprehensively depicted in Figure 2.12. This illustration highlights the progressive expansion and the transformative progression of scholarly investigations focused on the integration of machine vision and machine learning to enhance the accuracy and efficiency of pavement crack detection techniques.

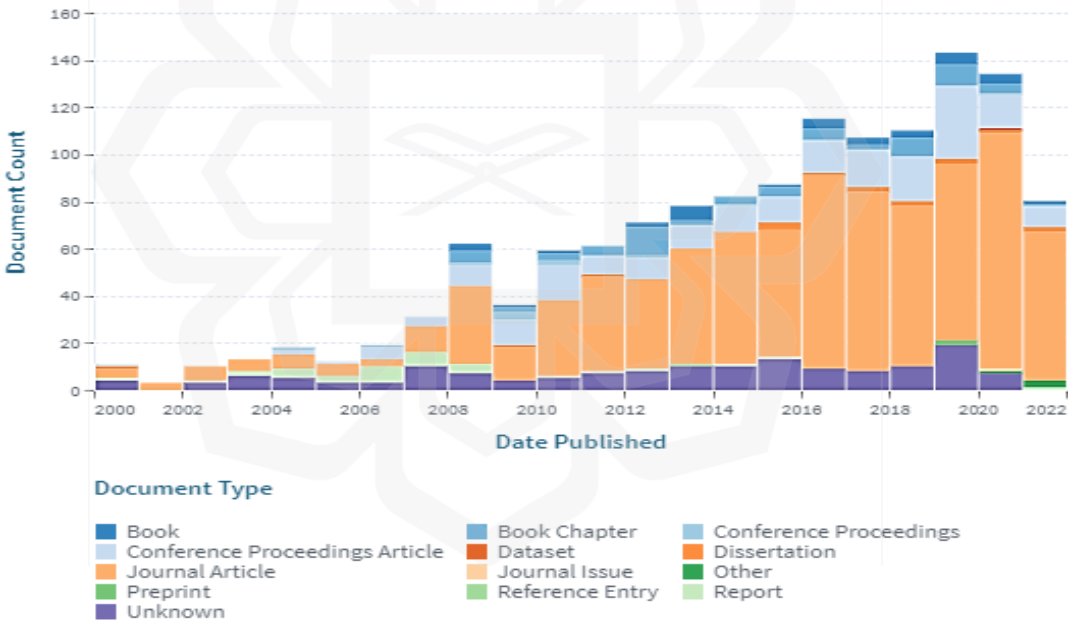



Figure 2.12. Scholarly Contributions on Pavement Crack Detection Utilizing Machine Vision and Machine Learning *(Accessed from Lens.org on 9/05/2022)*

The assessment of pavement surfaces traditionally relies on the subjective visual inspections by trained personnel and the quantitative analysis using specialized, often expensive, equipment. Visual inspections, while crucial, are marred by their labor-intensive nature, high costs, and a degree of subjectivity that could compromise

the consistency and reliability of the assessments, thereby potentially endangering the road infrastructure's integrity. This challenge is exacerbated in regions that lack the requisite sophisticated inspection tools, leading to less frequent evaluations of the road framework. In contrast, large-scale assessments employing mobile detection systems or laser scanning technologies are more systematically conducted. These methods utilize global positioning systems (GPS) to gather precise geospatial data from a moving vehicle, integrating GPS units with digital imaging, cameras, laser scanners, and omnidirectional video recorders. Although these quantitative inspections are accurate, their extensive nature incurs significant costs, making them less viable for smaller regions with limited financial resources.

To address these challenges, innovative approaches have been developed, combining in-vehicle cameras and image processing technologies to offer a more efficient and cost-effective means of examining pavement conditions. For example, previous research introduced an automated technique for detecting cracks in asphalt using image processing methods coupled with a Bayesian artificial intelligence approach (Q. Zou et al., 2019). Additionally, recent advancements have led to the development of pothole detection systems employing traffic cameras (W. Liu et al., 2019). The advent of deep neural networks has further enhanced the capability to accurately assess road surface damage. (A. Zhang et al., 2017) developed CrackNet, a system designed to predict class scores for individual pixels, thereby facilitating the detection of pavement cracks. However, many of these techniques primarily focus on identifying the existence of cracks rather than classifying them based on their types. Certain studies have attempted to categorize cracks by their orientation, such as vertical and horizontal (Alshandah et al., 2020), or by their patterns, such as longitudinal, transverse, and alligator cracking. Despite these efforts, the majority of research tends to classify cracks into a limited number of categories. It underscores the necessity for a more nuanced approach to accurately identify and classify the diverse types of road cracks to enhance real-world crack detection models.

An innovative unsupervised road crack detection method, leveraging the dark histogram and Otsu's thresholding technique, was introduced (Peng et al., 2015), yielding promising results particularly in scenarios with low signal-to-noise ratios. This approach was further refined through the use of crack width metrics, where

researchers applied an enhanced unsupervised learning algorithm based on the least squares method, effectively reducing errors in crack detection. A novel strategy was employed utilizing the concept of window's base power to segregate emerging cracks at various scales within the image, subsequently analyzing the interrelations among cracks of differing scales and developing a crack evaluation model grounded in multivariate statistical analysis.

To tackle the issue of irregular edge fractures possessing complex topological features, a novel pavement crack detection system was devised, based on a random forest algorithm. This model was calibrated by extracting fracture characteristics across multiple scales and orientations. In further advancements, a computational methodology was proposed for the detection of asphalt cracks (Sari et al., 2019), initiating with the pre-processing phase to smooth the surface and accentuate existing cracks. Support vectors were generated for each distinct segment using the Supervised Learning technique SVM, facilitating the differentiation of cracks. This method necessitates meticulous computational planning due to its complexity.

The realm of deep learning has seen substantial advancements across various domains of machine vision, including image classification, object detection, and segmentation. A plethora of deep learning-based algorithms, especially deep convolutional neural networks, have been developed for the specific purpose of road crack detection. These methodologies can be broadly categorized into three distinct approaches: image-based grouping, object detection-based, and pixel-level detection strategies. A single CNN model was utilized in (Y. Zhou et al., 2016) to approach the issue of asphalt crack detection as a multi-label classification challenge. Concurrently, DeepCrack was introduced (G. Li et al., 2022), employing an encoder-decoder architecture for the segmentation of asphalt images into crack and non-crack regions. Furthermore, a network structure was proposed (X. Wang et al., 2020), incorporating four convolutional layers and maximum pooling in the encoder phase to abstract features, followed by four subsequent modules in the decoder phase for detailed segmentation.

Figure 2.13 illustrates a comprehensive overview of AI-based crack detection models, categorizing them into machine learning and deep learning-based

frameworks. Initially, the field predominantly relied on supervised learning models within the machine learning paradigm, marking a significant progression from unsupervised models. However, the introduction of deep learning techniques for object detection has led to a paradigm shift, achieving superior detection outcomes. This integration of deep learning with traditional machine learning methods has given rise to hybrid models, blending the strengths of both approaches to enhance the efficiency and accuracy of pavement crack detection.

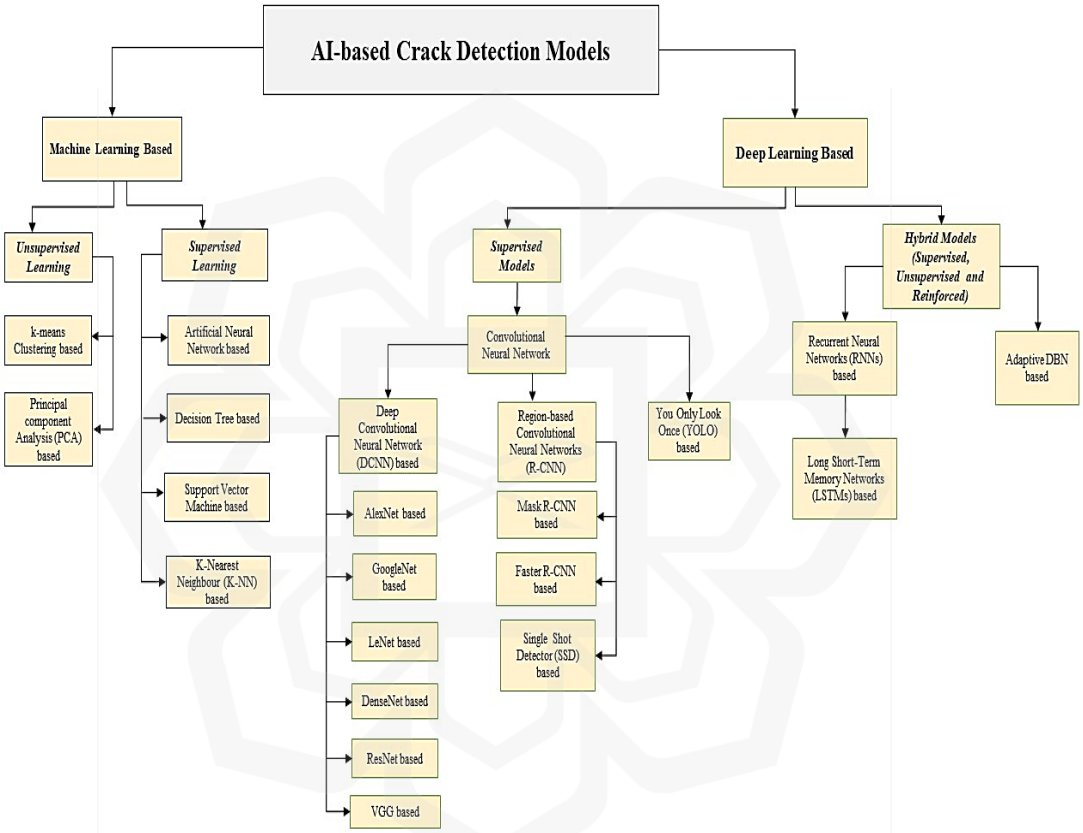



Figure 2.13. Categorization of AI-Based Models for Crack Detection

Considering the wealth of studies and the plethora of published works in the realm of identifying and categorizing pavement cracks, it is crucial to methodically compile, condense, and thoroughly evaluate this abundance of data. Table 2.5 compiles a curated selection of research within the area of pavement crack detection and categorization, shedding light on the approaches used, their advantages, and their intrinsic drawbacks.

Table 2.5. Synopsis of Selected Studies on Pavement Crack Detection

| Methodology | Strengths | Limitations | Citation |
|---|---|---|---|
| Convolutional Neural Network | Demonstrated promising model performance through data generalization and hybrid data training, with cross-testing enhancing the labeling process. Achieved 90% accuracy. | Does not account for the precise positioning of crack pixels, focusing solely on the binary presence or absence of cracks. | (Fan et al., 2018a) |
| Deep Convolutional Neural Network Fusion, U-Net model | Optimization of crack classification hyperparameters was achieved, with the model tested on a specially formulated dataset. The model also facilitates crack length calculation via pixel labeling and scanning. | The model's average accuracy was relatively low at 78.7%. | (Feng et al., 2020) |
| Deep Neural Networks | Utilized a self-created dataset, comprising 163,664 images collected in collaboration with local governments in Japan. | The method does not consider crack length, and the dataset's recall and precision were moderate, at 75%. | (Maeda et al., 2018) |
| Deep Supervision Learning | Utilized a feature pyramid and hierarchical boosting approach to improve low-level feature representation, markedly decreasing the time required for crack identification. | Evaluation focused primarily on time efficiency, with limited data on accuracy, recall, and precision. | (F. Yang et al., 2020) |
| Supervised Deep Convolutional Neural Network | Demonstrated superior performance to traditional techniques such as Boosting and SVM, with recall rates reaching 92.51%. | Relied on manual image annotation, raising concerns regarding cost-effectiveness and practical applicability. Crack length was not addressed. | (L. Zhang et al., 2016) |
| Convolutional Neural Network | Integrated ResNet with geographic information | Incidence of false positives suggesting | (Chun et al., 2021) |

| | | | |
|---|---|---|---|
| | systems, complemented by a mobile mapping system, achieving 94.3% accuracy. | cracks in images without actual cracks; the refinement of crack pixels was less than ideal. | |
| Wavelet decomposition, Ridgelet transform, Image enhancement | Proposed a novel image enhancement algorithm in ridgelet space, showing potential in enhancing and detecting road cracks. | The method, being more traditional, may not align with current technological expectations. | (Ashraf, Sophian, et al., 2022b) |
| SVM, Edge-Based Approach | Introduced an automated technique utilizing UAV-captured footage for crack detection, with key edge selection and ortho-image generation to exclude non-road elements. | The model's accuracy was relatively low at 75%. | (Dadrasjavan et al., 2019) |
| CNN, SVM | Achieved high precision in distinguishing between crack and non-crack images and classifying longitudinal cracks, with an accuracy of approximately 98%. | While accuracy is high, the model primarily focuses on simple detection rather than comprehensive crack classification. | (Bhat et al., 2021) |
| Deep Learning, Yolo v3 | Utilized Yolo v3 network training for crack detection, achieving an 88% accuracy rate. | Manual labeling was required, and the model did not significantly focus on crack classification. | (Nie & Wang, 2019) |

#### *2.3.7.3. Crack Characterization*

Beyond the mere detection of road cracks, the characterization of these imperfections plays a pivotal role in the maintenance and repair processes. Crack characterization involves not only identifying the nature of the crack but also measuring its dimensions. Historically, this process was predominantly manual, with road maintenance teams physically inspecting the cracks to ascertain their type and size, a procedure that proved both time-intensive and resource-heavy. Inspired by methodologies such as the Portuguese Trouble Catalog, a shift towards automated systems for crack characterization was proposed (Oliveira & Correia, 2013). Within

this context, the application of dual Gaussian models emerged as a notably effective technique for the unsupervised detection of cracks.

Upon the identification of a crack, a comprehensive asphalt crack inspection is initiated, as outlined in (X. Wang et al., 2020). This process has been validated through the analysis of asphalt surface images captured at various intervals, employing a multiscale constraint method that incorporates GPS-based coarse localization, image-level restriction, and metric restriction to accurately map original crack pixels onto the current crack image. These mapped pixels then serve as precise seed points for the subsequent investigation of the crack, with the Region Growing Method (RGM) being utilized to unearth newly formed cracks. The efficacy of this approach is underscored by an F-measure indicating an 88.9% improvement in crack analysis outcomes.

In a further development, the work presented in (Fan et al., 2020) introduces an innovative automated framework designed for the swift identification and categorization of cracks based on high-speed survey imagery. This method entails an initial pre-processing stage where morphological filters are applied to reduce pixel intensity variations, followed by the employment of a dynamic thresholding technique to isolate dark pixels indicative of potential cracks. Subsequently, the images are segmented into non-overlapping blocks, with entropy calculations used to refine the segmentation. A second round of dynamic thresholding applied to the entropy block matrix aids in pinpointing image blocks containing cracks, which are then classified according to their orientation (horizontal, vertical, miscellaneous) or identified as non-cracked. The robustness of this methodology has been tested using two distinct image databases, including one captured with advanced high-speed imaging equipment.

The integration of machine learning technologies into crack characterization processes has markedly improved the efficiency and accuracy of these assessments. By adopting block-level segmentation, cracks are isolated and subjected to type analysis, while pixel-level segmentation facilitates the determination of the crack's endpoints, thereby enabling precise measurement of the crack's dimensions. This advanced approach, bolstered by pixel-level segmentation, represents a significant leap forward in the domain of pavement crack analysis.

## 2.4. PERFORMANCE PARAMETERS

Performance parameters in deep and machine learning are quantitative metrics used to evaluate the effectiveness and accuracy of a model in solving a specific task. These metrics help measure the quality of predictions made by the model and provide valuable insights into its strengths and weaknesses. Understanding performance parameters is crucial for selecting the right model, tuning hyperparameters, and comparing different models for a given problem. Some of the essential performance parameters used in deep and machine learning:

1. **Accuracy:** Accuracy is one of the most fundamental performance metrics and is defined as the ratio of correctly predicted instances to the total number of instances in the dataset as shown in Equation 2.16. It provides a general measure of how well the model performs overall. However, accuracy can be misleading, especially when the classes are imbalanced, and the dataset has unequal class distribution.

$$Accuracy = Number\ of\ Correct\ Predictions\ /\ Total\ Number\ of\ Predictions \quad (2.16)$$

2. **Precision:** Precision is the proportion of true positive predictions (correctly predicted positive instances) to the total number of positive predictions made by the model as shown in Equation 2.17. It helps measure the model's ability to avoid false positives, indicating how precise the model is when it predicts a positive outcome.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive+False Positive}} = \frac{\text{True Positive}}{\text{Total Predicted Positive}} \quad (2.17)$$

3. **Recall (Sensitivity or True Positive Rate):** Recall is the ratio of true positive predictions to the total number of actual positive instances in the dataset. It quantifies the model's ability to capture all positive instances,

highlighting its sensitivity to detecting positive cases as shown in Equation 2.18.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive+False Negative}} = \frac{\text{True Positive}}{\text{Total Actual Positive}} \quad (2.18)$$

4. **Specificity (True Negative Rate):** Specificity is the ratio of true negative predictions to the total number of actual negative instances in the dataset. It reflects the model's ability to identify negative instances correctly and is particularly relevant when the dataset has imbalanced class distribution as shown in Equation 2.19.

$$Specificity = True\ Negatives\ /\ True\ Negatives + False\ Positives \quad (2.19)$$

5. **F1 Score:** The F1 score is the harmonic mean of precision and recall, providing a balanced measure of the model's accuracy. It considers both false positives and false negatives, making it a suitable metric for imbalanced datasets as shown in Equation 2.20.

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (2.20)$$

6. **Area Under the Receiver Operating Characteristic Curve (AUC-ROC):** The ROC curve is a graphical representation of the model's true positive rate (recall) against the false positive rate at various threshold settings. The AUC-ROC is the area under this curve, indicating the model's ability to discriminate between positive and negative instances across different thresholds. A higher AUC-ROC value implies a better-performing model.
7. **Mean Absolute Error (MAE) and Mean Squared Error (MSE):** MAE and MSE are performance metrics commonly used in regression tasks. MAE measures the average absolute difference between predicted and actual values, while MSE measures the average squared difference between predictions and actual values. Lower values of MAE and MSE

indicate better regression model performance. The equations are shown in 2.21 and 2.22.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_{true}^{(i)} - y_{pred}^{(i)}\right| \quad (2.21)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{true}^{(i)} - y_{pred}^{(i)}\right)^2 \quad (2.22)$$

8. **R-Squared (R2) Score:** The R-squared score quantifies how well a regression model fits the data compared to a simple mean baseline. It represents the proportion of variance in the target variable that can be explained by the model. Higher R-squared values indicate a better fit of the model to the data as shown in Equation 2.23.

$$R^2 = 1 - \frac{MSE}{Var(y_{true})} \quad (2.23)$$

9. **Mean Average Precision (MAP):** MAP is a performance metric often used in information retrieval and ranking tasks. It calculates the average precision across different recall levels, providing a comprehensive evaluation of the model's retrieval quality.
10. **Cohen's Kappa:** Cohen's Kappa is a performance metric used to evaluate the agreement between an automated model and human annotators in classification tasks. It accounts for the agreement that could occur by chance and provides a more reliable measure of model performance as shown in Equation 2.24.

$$\kappa = Po - Pe \,/\, 1 - Pe \quad (2.24)$$

where *Po* is the observed agreement and *Pe* is the expected agreement.

11. **Mean Absolute Percentage Error (MAPE):** MAPE is a metric used in forecasting tasks to measure the accuracy of predictions as a percentage. It calculates the average percentage difference between predicted and actual values, providing an understanding of the average error relative to the actual values.

12. **Root Mean Squared Error (RMSE):** RMSE is another common metric used in regression tasks. It is the square root of the average squared difference between predicted and actual values. RMSE is more sensitive to large errors compared to MAE and is widely used in cases where large errors should be penalized more.
13. **Confusion Matrix:** The confusion matrix is a table that presents the model's performance in a classification task. It shows the number of true positive, true negative, false positive, and false negative predictions, allowing a detailed analysis of the model's performance for each class.
14. **Categorical Cross-Entropy (Log Loss):** Categorical cross-entropy is a loss function commonly used in multi-class classification problems. It quantifies the dissimilarity between predicted probability distributions and true class labels. Lower cross-entropy values indicate a better-performing model.
15. **Mean Squared Log Error (MSLE):** MSLE is a metric used in regression tasks to measure the accuracy of predictions on a logarithmic scale. It calculates the average squared difference between the logarithms of predicted and actual values, making it less sensitive to large errors.
16. **Gini Coefficient and Lift Chart:** These metrics are commonly used in marketing and business analytics to evaluate model performance for ranking tasks. The Gini coefficient measures the inequality between predicted and actual rankings, while the Lift chart visually illustrates the model's effectiveness in identifying positive cases compared to random selection.
17. **Jaccard Index (Intersection over Union - IoU):** The Jaccard Index is used in segmentation and object detection tasks to evaluate the overlap between the predicted and ground-truth regions. It measures the similarity between two sets by calculating the size of their intersection divided by the size of their union.
18. **Balanced Accuracy:** Balanced accuracy is particularly useful for imbalanced datasets. It calculates the average accuracy of each class, providing a balanced measure that considers both majority and minority classes as shown in Equation 2.25.

$$Balanced\ Accuracy = Sensitivity + Specificity/2 \quad (2.25)$$

19. **Matthews Correlation Coefficient (MCC):** MCC is a metric that takes into account true positive, true negative, false positive, and false negative predictions in binary classification tasks. It provides a balanced measure of classification performance, even for imbalanced datasets.
20. **Explained Variance Score:** The explained variance score is used in regression tasks to quantify how well the model accounts for the variance in the target variable. It represents the proportion of variance explained by the model relative to the total variance in the target variable.

## 2.5. EVALUATION AND COMPARISON OF VARIOUS CRACK DETECTION AND CLASSIFICATION APPROACHES

The evaluation of crack detection and classification models hinges on their ability to accurately identify the presence of cracks and categorize their types, as delineated in various studies. Key metrics such as accuracy, precision, sensitivity, and the F1 score—a balance between precision and recall—are essential for this assessment. However, the performance of these metrics is often adversely affected by the skewed distribution of data within the datasets used for model training, posing a challenge to the integrity of the evaluation process (Ashraf et al., 2020). To construct an effective model, considerations must extend beyond data collection and preprocessing to include the fine-tuning of algorithmic hyperparameters and ensuring the model's generalizability. The confusion matrix serves as an indispensable tool in this context, facilitating the computation of accuracy and precision by categorizing outcomes into True Positives, True Negatives, False Positives, and False Negatives, as illustrated in Figure 2.14.

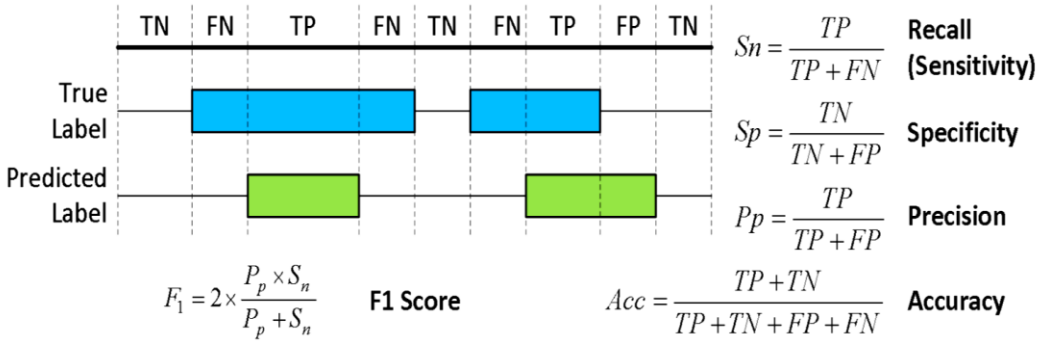



Figure 2.14. Performance metrics in machine learning

In the exploration of performance metrics central to machine learning, particularly within the context of crack detection and classification from images and videos, a variety of neural network approaches and deep learning algorithms have been employed. Techniques such as Convolutional Neural Networks, Modular Neural Networks (Gunawan et al., 2018), Residual Depthwise Separable CNNs (Ihsanto et al., 2020), Recurrent Neural Networks, alongside methods like Yolo, SSD, ResNet, UNet, and LeNet, represent the forefront of this research area. The evaluation of these methods, as showcased in Table 2.6, provides a comparative analysis based on performance parameters, illustrating the advancements and outcomes of scholarly work in this field.

Table 2.6. Performance of Various Machine Learning Models

| **Technique** | **Performance Metrics** | **Reference** |
|---|---|---|
| SVM<br>BOOSTING<br>CONV NETS | Precision: 0.81, 0.73, 0.86<br>Recall: 0.67, 0.75, 0.92<br>F1 Score: 0.73, 0.74, 0.89 | (L. Zhang et al., 2016) |
| Convolutional Neural Network | Precision: 0.911; Recall: 0.948; F1 Score: 0.922 | (Fan et al., 2018b) |
| GoogleNet CNN, FPN | Precision: 0.80(Mohan & Poobal, 2018b)e: 0.81 | (Mohan & Poobal, 2018b) |

| | | |
|---|---|---|
| Random Structured Forests, SVM | Precision: 0.96 | (Olson et al., 2018) |
| CNN | Precision: 0.8696; Recall: 0.9251; F-Measure: 0.8965 | (Bhat et al., 2021) |
| Deep Convolutional Encoder-Decoder Network | Recall: 0.71; Precision: 0.77; Intersection of Union: 0.59 | (Han et al., 2018) |
| CNN | Precision: 0.90; Recall: 0.87; F-Measure: 0.88 | (Sari et al., 2019) |

## 2.6. CHALLENGES

In recent years, the field of pavement crack detection and classification has gained prominence, particularly at the intersection of Human-Computer Interaction (HCI) and image analysis. The surge in research activities in this domain reflects an acute need for more refined technological solutions. By bridging the gap between civil engineering principles and HCI methodologies, this collaborative research effort aims to tackle the complexities involved in accurately identifying and categorizing cracks in pavement surfaces.

A primary obstacle in this field is the procurement of high-quality data, which plays a pivotal role in the development and performance of machine learning models. The reliance on publicly available datasets for research often leads to models that underperform when applied to real-world scenarios. This highlights the necessity of prioritizing data that closely mirrors real-world conditions in the development of models for crack detection and classification.

Moreover, the quality of input data, be it images or videos, is frequently compromised by noise factors such as shadows, variations in pixel intensity, and different types of noise, all of which can significantly impede model performance. Despite numerous pre-processing efforts to mitigate these issues, there remains a need for enhanced data refinement through advanced digital image processing techniques.

The process of reducing data dimensionality and selecting pertinent features also presents significant challenges due to the complexity and cost associated with choosing the most appropriate subset of features from a vast pool. Convolutional Neural Networks (CNNs) are lauded for their ability to generate new features from existing ones, thereby condensing the overall feature set and effectively encapsulating the original data attributes.

The efficacy of crack detection and classification systems is heavily dependent on the classifier's ability to accurately interpret algorithmic outputs. The training of conventional machine learning classifiers, including CNNs, can be exceedingly time-consuming, particularly with techniques like max pooling. Additionally, the computational demands for traditional classifiers such as k-Nearest Neighbors (kNN), Decision Trees, and Support Vector Machines (SVM) escalate with the expansion of the dataset. To surmount these hurdles, a variety of deep learning models and architectures like ResNets, U-Nets, YOLO, DenseNets, and SSDs are increasingly being adopted.

The journey of developing a comprehensive model for crack detection, classification, and characterization is fraught with challenges at each stage, from initial data collection to data annotation, model construction, and ultimately, model evaluation. A meticulous approach to each phase of development is essential to minimize errors and achieve promising outcomes during the evaluation stage.

## 2.7. SUMMARY

The endeavour to automate the identification of pavement cracks has attracted significant interest for its real-world relevance, transitioning from classic image processing to sophisticated AI and deep learning strategies. This section presents a detailed examination of the latest techniques put forward and employed by scholars in recognizing, categorizing, and detailing pavement cracks. It examines the various crack forms present in both concrete and asphalt surfaces, providing insights into numerous studies, their methodologies, and outcomes.

This segment introduces a standard structural blueprint for detecting, categorizing, and detailing cracks, utilizing image analysis and machine learning approaches. It further discusses the varied data gathering methods and image types employed in this area. Acknowledging the pivotal importance of data for the efficacy of model creation, this part offers a synopsis of accessible data compilations, including detailed descriptions, to support model training and evaluation.

Additionally, this segment conducts a comparative review of different models crafted for crack detection and categorization, assessing them against essential efficiency indicators. It also highlights ongoing obstacles encountered by scholars in this field. By the end of this chapter, Objective 1, which encompasses a thorough understanding, identification and analysis of the current state of research in pavement crack recognition, is fulfilled.

# CHAPTER THREE

# METHODOLOGY

## 3.1. INTRODUCTION

In this study, a comprehensive approach to road crack detection and characterization is undertaken through the development of three synergistic models. The first, a customized adaptation of the YOLOv7 architecture, is tailored to meet the specific needs of our dataset and project objectives, optimizing its capabilities in object detection. Progressing further, the second model utilizes the advanced features of the YOLOv8x algorithms, focusing on high-precision object detection, segmentation, and classification. This model is fine-tuned to ensure peak performance within our research parameters. The final model, a hybrid, ingeniously combines the sophisticated pattern recognition of deep learning with the reliability of conventional algorithms, creating a potent synergy that leverages the strengths of both approaches. This integrative modeling strategy not only addresses the multifaceted challenges of automated crack analysis but also aligns with the specific objectives of this research, paving the way for significant advancements in the field of road infrastructure analysis.

## 3.2. MODEL 1: CUSTOMIZED YOLOV7 FOR CRACK DETECTION AND CLASSIFICATION

In this research, we present a novel model for the identification and classification of pavement cracks. This methodology utilizes two distinct data sources: firstly, road condition data captured via a camera-equipped survey vehicle, and secondly, publicly available imagery of road cracks. This data is subjected to a series of advanced preprocessing steps, including the extraction of image frames, labeling, augmentation, and resizing. Subsequently, the processed data is segmented into training, testing, and validation sets. These datasets are then analyzed using the YoloV7 algorithm, an advanced and tailored approach for this specific use case. Following the algorithmic

analysis, the model's effectiveness is rigorously evaluated. The proposed approach is depicted in Figure 3.1.

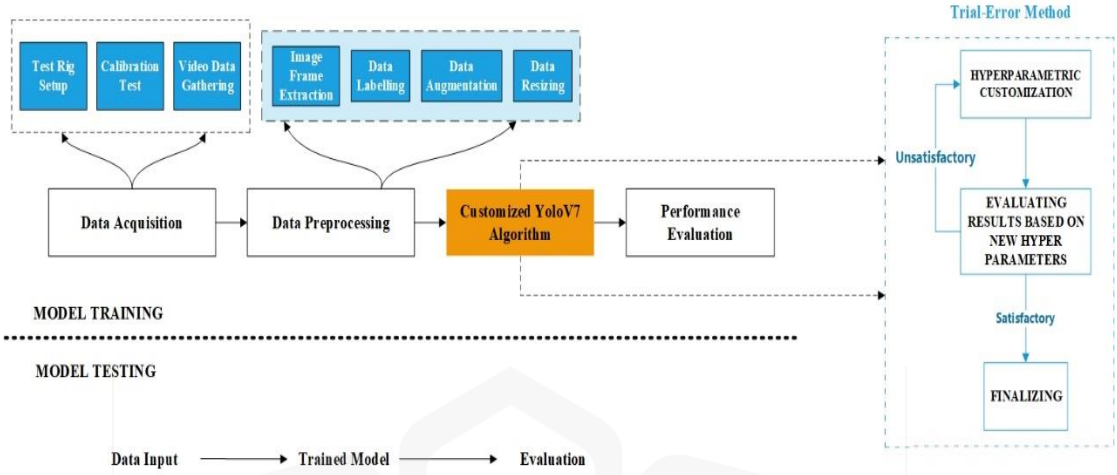



Figure 3.1. Proposed Research Methodology

### 3.2.1. Data Acquisition

This study utilizes the RDD2022 (Arya et al., 2022) online dataset, an extensive collection of 47,420 road photographs from six diverse countries: Japan, India, the Czech Republic, Norway, the United States, and China. The dataset includes over 55,000 instances of documented road damage. Four types of road damages, namely transverse cracks, longitudinal cracks, alligator cracks, and potholes, are comprehensively represented in this dataset. Notably, for this investigation, we extracted data captured via smartphones affixed to motorcycles from the RDD2022. Furthermore, our study enriched the dataset with real-world observations, meticulously gathering images of road cracks from the vicinity of the International Islamic University Malaysia, Gombak campus. This specialized dataset was obtained using a GoPro Hero 8 camera mounted on a designated inspection vehicle. Figure 3.2 offers a visual representation of the vehicle-mounted camera system employed for data collection.

Figure 3.2. Installed Camera Setup on the Survey Vehicle

The data acquisition process was carefully tailored to simulate real-life road environments. This entailed adjusting the camera's height and orientation relative to the surface to span a 3.1m width of the road, aligning with the standard width of a single-lane roadway. Such calibration aimed to capture a view that accurately reflects common road conditions. The intricacies of the camera specifications and the outcomes of the calibration efforts are detailed in Tables 3.1 and 3.2. For collecting this specific dataset, Calibration Setup 2 was selected, positioning the camera at a height of 1.6m.

Table 3.1. Specifications of the Camera Configuration

| **Attribute** | Details |
|---|---|
| **Camera Model** | GoPro Hero 8 |
| **Mounting Hardware** | GoPro Rod Mount |
| **Operating Mode** | Custom Video Setting |
| **Image Resolution** | 1080p |
| **Frame Rate Options** | 24 fps, 60 fps |
| **Lens Type** | Linear Field of View |
| **Video Bit Rate** | Standard Quality (45 Mbps) |
| **Minimum ISO Setting** | 100 |

Table 3.2. Outcomes of Camera Calibration

| Calibration Setup | Road Coverage Width | Camera Orientation to Ground | Proximity to Marked Road Segment | Installation Height |
|---|---|---|---|---|
| **Setup 1** | 3.1 meters | 90° ± 35° | 1.1 meters | 1.30 meters |
| **Setup 2** | 3.1 meters | 90° | Directly above midpoint | 1.6 meters |

### 3.2.2. Data Preprocessing

Before the collected data was utilized for training the deep learning model, it was subjected to a series of customized preprocessing procedures. These steps involved extracting still frames from video recordings, annotating these images, augmenting the dataset, and adjusting the image sizes for consistency.

The data specific to this research was initially recorded as video footage. From these recordings, individual frames were extracted to serve as separate data points. Each of these frames underwent manual labeling, employing the Roboflow annotation tool designed for precise data marking. In addition, the RDD2022 public dataset, which includes pre-annotated images of road cracks, was incorporated into the study. The labeling conventions used for both the newly collected data and the RDD2022 images are comprehensively outlined in Table 3.3.

Table 3.3. Annotation Labels

| Label Category | RDD 2022 Labels | Custom Data Labels |
|---|---|---|
| Longitudinal Crack | D00 | crack_long (1) |
| Transverse Crack | D10 | crack_trans (2) |
| Alligator Crack | D20 | crack_alligator (0) |
| Potholes | D40 | Pothole (3) |

The accuracy of forecasts produced by Supervised Deep Learning models is significantly influenced by the quantity and variety of the data used for training. Nevertheless, a prevalent challenge in crafting deep learning models is the scarcity of extensive and varied datasets. Data augmentation seeks to address this challenge by enhancing the dataset's richness, either by applying subtle modifications to existing data or employing machine learning techniques to create new data points. In this study, the following data augmentation techniques were applied:

1. **Blurring:** This effect softens the edges of elements in an image, making them appear slightly unfocused. In our dataset augmentation, we applied a 2px blur to the RDD2022 images and a 4.5px blur to the images from our custom dataset.
2. **Brightness:** This pertains to the light intensity or darkness of an image. The brightness level of an image, post-capture or digital conversion, reflects its light intensity. In our augmentation process, we adjusted the brightness levels for images from both datasets within a range of +20% to -20%.

Following the augmentation, we resized the image samples, a vital step in image preprocessing for computer vision tasks. Resizing alters the dimensions of an image, for example, scaling down from 1920×1080 pixels to 480×270 pixels, which can significantly reduce the computational load and time required for training deep learning models. For this project, we standardized the size of images from both datasets to 640×640 pixels.

Table 3.4 provides a comprehensive overview of the total number of data samples utilized in this study, both before and after applying the preprocessing techniques. The datasets were segmented into training, validation, and testing subsets, following an 80:20 split. Within this allocation, the testing and validation subsets each comprised 10% of the total data.

Table 3.4. Image Sample Counts Pre and Post Preprocessing

| Dataset | Count Before Preprocessing | Count After Preprocessing |
|---|---|---|
| RDD2022 | 2477 | 3893 |
| Custom Data | 470 | 851 |

## 3.3. CUSTOM YOLOV7 DEEP LEARNING MODEL

Deep learning, a sophisticated subset of machine learning, draws its inspiration from the human brain's architecture and operational principles, forming what are known as artificial neural networks. It's a methodology that enables machines to learn from examples in a manner akin to human learning, a field that has garnered considerable interest due to its groundbreaking achievements. Deep learning systems are adept at directly absorbing knowledge from data in various forms, such as images, text, or audio, making them highly effective for tasks involving detection and classification. Remarkably, certain deep learning-based models have achieved extraordinary levels of accuracy, sometimes surpassing human performance. These models are developed through the use of large datasets with labeled instances and the application of complex layers within neural networks.

In the context of this research, we utilized the YOLOv7 algorithm (C.-Y. Wang et al., 2022), a part of the “You Only Look Once” lineage known for its efficiency in real-time object detection. YOLO models, categorized as single-stage object detectors, consist of three main structural elements: the backbone (spine), neck, and head. The backbone processes the input image frames, the neck integrates and refines the features extracted from the backbone, and the head uses these features to predict object locations and classifications, marking them with bounding boxes. YOLO employs techniques like non-maximum suppression (NMS) during post-processing to finalize its predictions. Figure 3.3 illustrates the core architecture of the YOLO model, showcasing its innovative design.

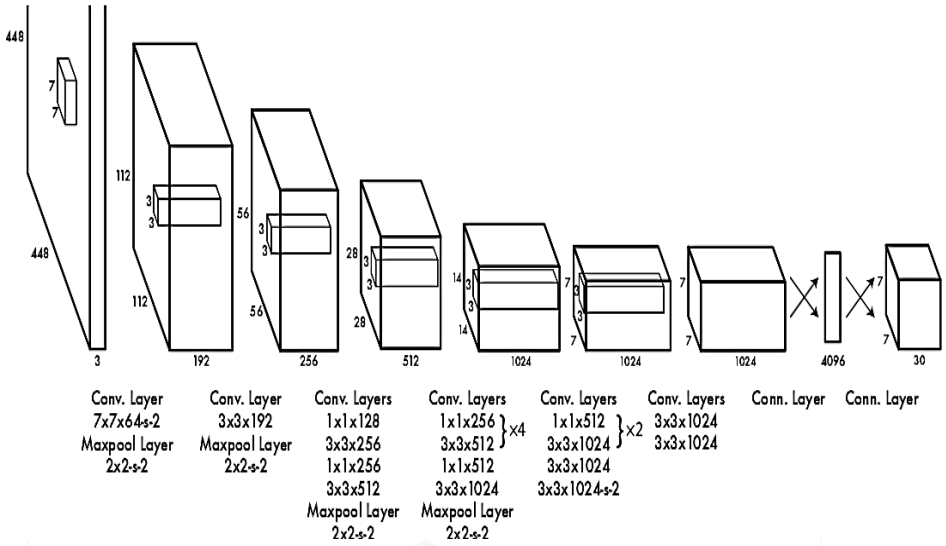



Figure 3.3. YOLO Architecture (Redmon et al., 2015)

Within this study, the YOLOv7 model was trained on preprocessed images, each standardized to a resolution of 640×640 pixels. The algorithm was finely tuned to meet the specific requirements of our research by adjusting a range of hyperparameters. The details of these tailored configurations are summarized in Table 3.5.

Table 3.5. Tailored Hyperparameter Settings

| **Hyperparameter** | **Value** |
|---|---|
| Batch Size | 20 |
| Epochs | 15 (for RDD2022)<br>50 (for Custom Dataset) |
| Initial Learning Rate | 0.01 |
| Final Learning Rate | 0.1 |
| Weight Decay | 0.0005 |
| Box loss gain | 0.05 |
| Cross Entropy Loss | 0.3 |
| Momentum | 0.9 |

## 3.4. MODEL 2: CUSTOMIZED YOLOV8X-SEG MODEL FOR CRACK DETECTION AND CHARATERIZATION

The research methodology for this study, illustrated in Figure 3.4, establishes a novel approach for the detection and characterization of road cracks. It initiates with an innovative data acquisition process that meticulously calibrates camera settings to accurately capture road images, which sets a new standard in data gathering precision.

Next, data collection is undertaken from various roads to compile a diverse and comprehensive dataset for analysis, broadening the scope of study beyond what previous research has encompassed. This thorough data collection process also includes the transformation of videos into individual frames to better enable detailed examination.

The study further stands out for its incorporation of a variety of data augmentation techniques, including flipping, contrast conversion, random brightness adjustment, and random exposure adjustment, to effectively enhance and diversify the dataset. The data labeling process adopted for instance segmentation is another novel feature of this research, enabling precise demarcation and localization of road cracks that contributes significantly to detection accuracy.

In an important move, the YOLOv8x architecture is tailored to align specifically with the crack detection requirements, enhancing its efficiency in detecting, segmenting, and classifying road cracks. Post segmentation, the study introduces an innovative process of converting segmented images into binary format based on the masks. This unique approach effectively separates cracks from the background, allowing for precise measurement of crack sizes.

Finally, the performance evaluation of the model provides a rigorous assessment of its effectiveness and accuracy in detecting and characterizing road cracks, showcasing the model's superior predictive capabilities. Taken together, these groundbreaking aspects position this research methodology as a trailblazer in the field of road crack detection and characterization.

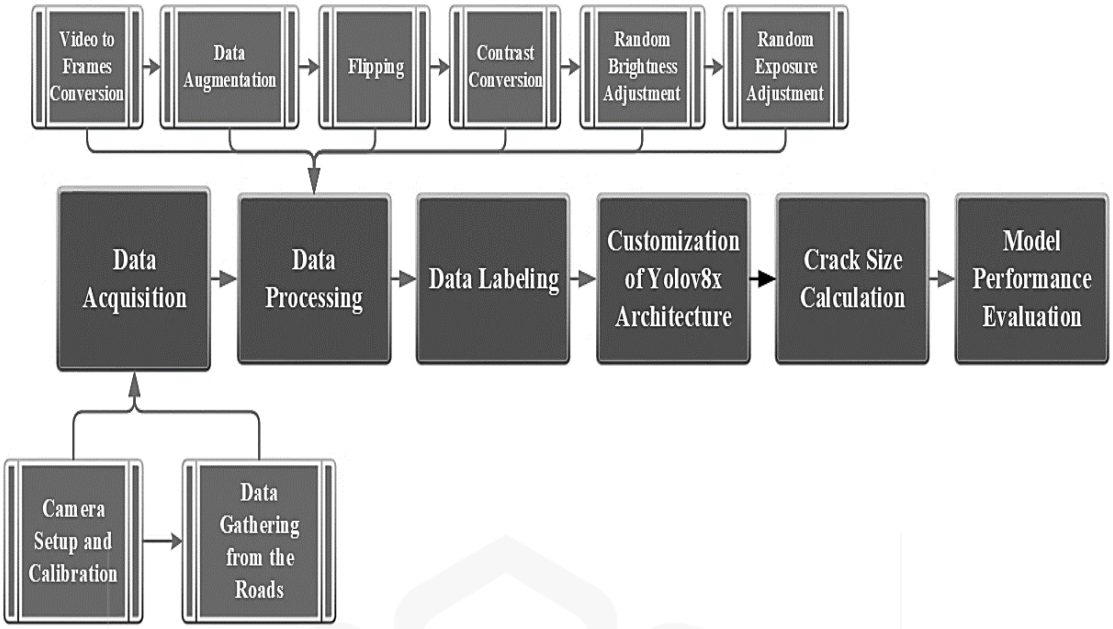



Figure 3.4. Process diagram of the executed work

### 3.4.1. Data Acquisition

In deep learning, comprehensive data collection is paramount for robust model training and validation. Accurate and diverse datasets underpin the efficacy and reliability of deep learning algorithms (Ashraf, Gunawan, et al., 2022). In the pursuit of innovative road analysis, the data acquisition phase of this research took an unprecedented approach to capture the intricacies of roads around Selangor and Kuala Lumpur. Utilizing a GoPro Hero 8 camera, the research team engineered a specialized mounting setup affixed to an inspection vehicle, a novel combination designed specifically for this study.

What sets this data collection apart is the meticulous design that mirrors real-world conditions, a critical aspect often overlooked in similar research. The camera's height and angle were carefully calibrated to cover a road width of 3.1 meters, reflecting the average width of a single lane on typical roads. This deliberate alignment not only ensures a more realistic representation but also advances the understanding of typical road scenarios, a groundbreaking development in the field.

The intricacies of the camera specifications and the outcomes of the calibration efforts are detailed in Tables 3.6 and 3.7. For collecting this specific dataset, Calibration Setup 2 was selected, positioning the camera at a height of 1.6m.

Table 3.6. Specifications of the Camera Configuration

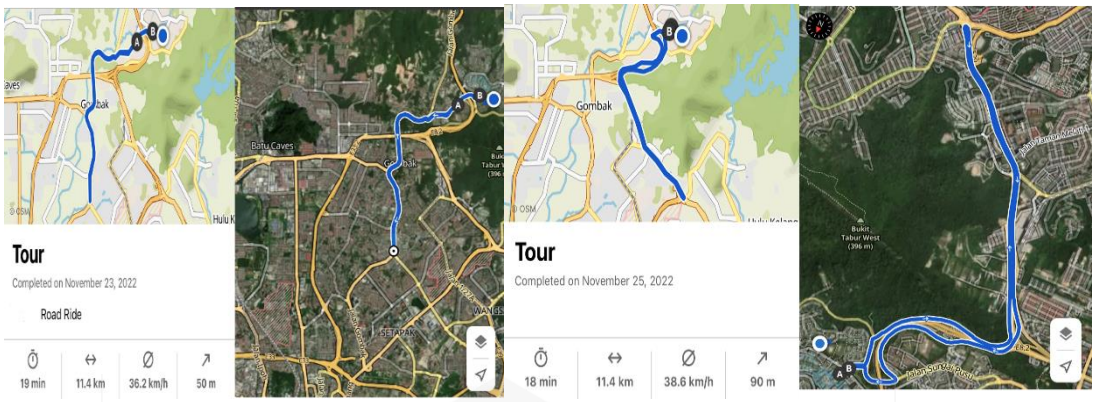
| **Attribute** | Details |
|---|---|
| **Camera Model** | GoPro Hero 8 |
| **Mounting Hardware** | GoPro Rod Mount |
| **Operating Mode** | Custom Video Setting |
| **Image Resolution** | 1080p |
| **Frame Rate Options** | 24 fps, 60 fps |
| **Lens Type** | Linear Field of View |
| **Video Bit Rate** | Standard Quality (45 Mbps) |
| **Minimum ISO Setting** | 100 |

Table 3.7. Outcomes of Camera Calibration

| **Calibration Setup** | **Road Coverage Width** | **Camera Orientation to Ground** | **Proximity to Marked Road Segment** | **Installation Height** |
|---|---|---|---|---|
| **Setup 1** | 3.1 meters | 90° ± 35° | 1.1 meters | 1.30 meters |
| **Setup 2** | 3.1 meters | 90° | Directly above midpoint | 1.7 1.6 meters |

Complementing the data acquisition methodology detailed earlier, the research further encompasses a detailed mapping of the areas from where the road data was collected around Selangor and Kuala Lumpur. This is vividly illustrated in Figures 3.5(a, b), which present the GPS road maps of the selected regions. The inclusion of these maps not only enhances transparency and reproducibility but also offers a spatial

context that allows future researchers to understand the diversity and specificity of the locations.




(a) (b)

Figure 3.5. (a) GPS road Data around Selangor, (b) GPS road Data around Kuala Lumpur

### 3.4.2. Data Preprocessing

Data preprocessing in machine learning and deep learning ensures that models receive quality input, directly influencing their performance and accuracy. By cleaning, normalizing, and transforming raw data, biases and irregularities are eliminated, laying a solid foundation for effective algorithm training. Proper preprocessing not only accelerates convergence but also enhances the generalizability and interpretability of models (Gawhade et al., 2022). In the intricate process of data preprocessing for our research, several methodical and advanced techniques were employed to fine-tune the quality of data, ensuring it was primed for a rigorous road analysis. Initially, we took videos that were captured using a vehicle mounted camera setup and segmented them into individual frames. This pivotal step transitioned continuous visual information into distinct and analyzable image units, offering a clearer granularity of detail.

For the sake of uniformity and to cater to the specific requirements of deep learning frameworks, each of these frames was resized to a consistent 640x640 pixel resolution. This uniform dimensioning not only eradicated variations but also optimized the images for computational efficiency in subsequent processing stages. Understanding the importance of a varied dataset in training robust models, we

introduced diversity by applying horizontal and vertical flips. Given a 50% probability of occurrence for each type of flip, this strategy enriched our dataset, simulating an expansive array of perspectives and real-world road scenarios, making the model more robust and versatile.

Furthermore, we recognized the importance of lighting conditions in road analysis. As such, we systematically adjusted parameters such as image brightness and exposure. Specifically, the brightness of each image was tweaked within a spectrum ranging from -40% to +40%, and exposure saw modulations between -16% and +16%. These calibrations were not arbitrary; they were devised to replicate the myriad lighting conditions a vehicle might encounter, from the low-light conditions of dusk or an overcast day to the high-exposure scenarios of noon or a sunlit highway. Through this meticulous preprocessing regimen, our aim was to cultivate a dataset that wasn't just vast, but also representative of the multifaceted conditions of real-world roads. This would, in turn, provide a solid foundation for training a machine learning model with enhanced accuracy and versatility.

### 3.4.3. Data Labeling

Data labeling is the bedrock of supervised machine learning, transforming raw information into a format algorithms can learn from. By providing a clear framework of input-output relationships, it guides models to discern patterns and make accurate predictions (Dong & Rekatsinas, 2018). In essence, without accurate data labeling, even the most advanced machine learning systems would lack direction and purpose. In this research, we used the Roboflow data annotation tool to label the frames extracted from recorded videos, focusing on three specific classes of road cracks: alligator cracks, longitudinal cracks, and transverse cracks. The labels assigned to these classes were "crack-alligator," "crack-long," and "crack-trans," respectively. This methodical approach to labeling, based on instance segmentation, ensures precise identification and categorization of the most common types of road damage.

The images were strategically split so that 80% were used for training and the remaining 20% for validation and testing. This division, chosen to provide a

comprehensive learning base while also allowing for effective validation and testing, was organized as shown in Table 3.8.

Table 3.8. Summary of Data Distribution for Training, Validation, and Testing of Road Crack Classes

| **Class** | **Label** | **Training Images** | **Validation Images** | **Testing Images** | **Total** |
|---|---|---|---|---|---|
| Alligator Cracks | crack-alligator | 760 | 80 | 40 | 880 |
| Longitudinal Cracks | crack-long | 760 | 80 | 40 | 880 |
| Transverse Cracks | crack-trans | 380 | 47 | 23 | 450 |
| **Total** | | **1900** | **207** | **103** | **2210** |

With a total of 1900 training images, it was ensured a robust learning experience for the model was attained. Additionally, the 207 validation images and 103 testing images allowed for a thorough evaluation of the model's performance. Some representational images of annotated data are shown in figure 3.6.

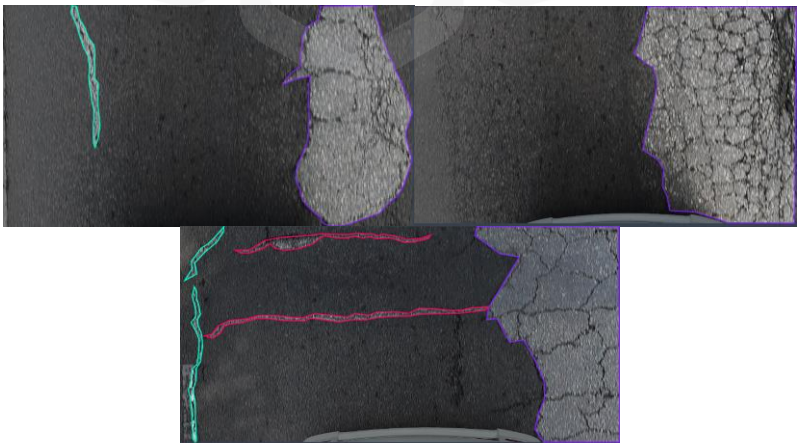


Figure 3.6. Annotated image frames with crack type for instace segmentation

### 3.4.4. Yolov8x Model Customization

The architecture of YOLOv8x is carefully designed to efficiently learn from both detection and segmentation tasks. It includes a backbone for hierarchical feature learning and a specialized head for segmentation prediction. The detection part provides bounding boxes and class probabilities as earlier YOLO versions, while the segmentation part extends these predictions to pixel-level accuracy. In the customization of the YOLOv8x-seg architecture for this research, a key modification was made by adding mask branch, adding a convolutional layer after the last classification layer. The convolutional layer had the same number of output channels as the number of classes that were been detected. The decision to add the mask branch to the head, rather than the backbone, stemmed from a careful analysis of their respective responsibilities and the implications for the model's performance.

Within the YOLO architecture, the head is tasked with predicting the bounding boxes of objects, while the mask branch specializes in predicting the pixel-level masks for these objects. Integrating the mask branch with the head fosters a more nuanced understanding of the relationship between the bounding boxes and the masks. This synergy leads to better performance in mask prediction. Conversely, adding the mask branch to the backbone, responsible for feature extraction, would not align well with the nature of the mask prediction task. While feasible, this approach is less effective as it does not leverage the specialized capabilities of the head in object localization. The algorithm presented in Figure 3.7 presents more understanding on the model customization and development.

```
Initialize:
    model = initialize_model('YOLOv8x-seg')
    Customize model architecture:
    add_mask_branch(model, head_layer)
    add_conv_layer_after_classification(model,
    num_output_channels=len(crack_types))
For each epoch:
    loss = train_model(model, labeled_data)
    If loss < loss_threshold:
```

```
                    break
        evaluate_model(model, validation_data)
                    Initialize:
    crack_types = ['longitudinal', 'transverse', 'alligator']
      classified_cracks = classify_cracks(model, frames,
                        crack_types)
                characterized_cracks =
        characterize_cracks(classified_cracks,
          method='pixel_level_segmentation')
analyze_crack_properties(characterized_cracks, metrics=['size',
                'shape, 'orientation'])
                    For each epoch:
                adjust_model_parameters(model)
    accuracy, recall = validate_model(model, validation_data)
      If accuracy > accuracy_goal and recall > recall_goal:
                        Break
                    Initialize:
    real_world_performance=test_model(model,real_world_condit
                        ions)
     benchmark_results = benchmark(model, existing_solutions)
      print('Model performance:', real_world_performance)
      print('Benchmark results:', benchmark_results)
```

Figure 3.7. Proposed YOLOv8x-Seg Algorithm for Pavement Crack Detection and Segmentation

In the endeavor to create an adept model for not only detecting but also characterizing road cracks using instance segmentation, hyperparameter customization emerged as a paramount step. This process required a detailed and explorative trial-and-error approach. Given the multifaceted nature of road crack data, which mirrors real-world conditions and involves complex patterns, a specialized approach was necessary. The goal was to not merely identify the cracks but to segment and characterize them according to specific classes, namely crack-alligator, crack-long, and crack-trans. Instance segmentation enabled the precise localization of road cracks and allowed for a detailed understanding of their type, shape, and orientation.

The hyperparameter tuning, conducted through a methodical process of trial and error, targeted the unique complexities of this task. It led to the discovery of the optimal combination of hyperparameters, thereby enhancing the model's ability to recognize and analyze the multifarious features present in road crack data as shown in Table 3.9.

Table 3.9. Customized Hyperparameters

| **Hyperparameter** | **Value** |
|---|---|
| Initial Learning Rate (lr0) | 0.01 |
| Final Learning Rate (lrf) | 0.1 |
| Weight Decay | 0.0005 |
| Box loss gain (box) | 7.5 |
| Class loss gain (cls) | 0.5 |
| Dual Focal Loss (dfl) | 1.5 |
| Keypoint Object Loss Gain (kobj) | 1.0 |
| hsv-h | 0.015 |
| hsv-s | 0.7 |
| hsv-v | 0.4 |
| Momentum | 0.937 |

### 3.4.5. Crack Size Assessment

A critical aspect of the methodology revolves around the assessment of the size of detected cracks. The steps, though intricate, ensure accurate estimation of crack dimensions, vital for infrastructure maintenance planning.

1. Segmentation Output: The YOLOv8x-seg model, trained and optimized on the collected dataset, produces a segmented image as its output. This

image distinctly marks the cracks and classifies them based on their types: alligator, longitudinal, or transverse.

2. Image Cropping: For a focused and accurate assessment, the segmented images were cropped to isolate and retain only the crack regions. This cropping eliminates unnecessary parts of the image, ensuring a more precise calculation of the crack's size. Furthermore, by narrowing down the region of interest, computational costs were minimized, and potential noise from other parts of the image was eradicated, leading to more reliable results.

3. Binary Conversion: Leveraging computer vision and image processing techniques, the segmented output is transformed into a binary format. The segmented region highlighting the crack displays pixels of a particular kind, distinct from the rest of the image. This distinction arises from the fact that the segmented cracks in the model's output are characterized by a unique color. Using this color as a reference, the image is converted to a binary format where the crack region is represented by one color (usually white) and the non-cracked regions by another (typically black).

4. Pixel Calculation: Once in binary format, the crack pixels are enumerated based on Equation 3.1. This quantified representation offers a direct measure of the extent of road damage.

$$C_p = T_{wp} \qquad (3.1)$$

Where$C_p$ are the Crack Pixels and $T_{wp}$ are the total number of white pixels in the Binary Image

5. Size Conversion: To translate the crack size from pixels to real-world measurements, a conversion formula is utilized. Given the camera's height from the road (1.6m) and the road width coverage (3.1m), the relationship between the pixels and real-world dimensions can be defined by:

For Transverse and Longitudinal Cracks: The relationship between the pixels and real-world length is defined by Equation 3.2:

$$C_l = C_p \times (3.1\ meters\ /\ Total\ width\ n\ pixels) \qquad (3.2)$$

Were, $C_l$ *is the* Crack Length (in meters).

For Alligator Cracks: Considering the same parameters, but since the alligator crack covers an area, the relationship between the pixels and the real-world area is defined by Equation 3.3:

$$Crack\ Area\ =\ C_p\ \times\ (3.1\ meters\ \times\ C_f\ /\ Total\ pixels\ in\ the\ image) \quad (3.3)$$

Where $C_f$ is Camera's Field of View in the transverse direction ($in\ meters$) which refers to the length of the road that the camera captures in its frame from its given height of 1.6m.

The outlined approach for crack size assessment offers several advantages. First, it ensures a systematic and scalable method to gauge road damage. By harnessing the power of instance segmentation combined with traditional image processing, it bridges the gap between high-tech AI models and tangible, on-ground metrics. This integration is crucial for operationalizing AI in the domain of infrastructure inspection, rendering the methodology both novel and practically significant.

## 3.5. MODEL 3: ADVANCED HYBRID DEEP LEARNING FOR CRACK DETECTION, CLASSIFICATION AND SEGMENTATION

This section unfolds the methodological architecture of a novel deep learning model meticulously designed for the segmentation and subsequent classification of road cracks as shown in Figure 3.8. The approach encapsulates a sequential pipeline that initiates with segmentation to precisely localize the cracks, followed by classification to categorize their characteristics. This dual-phase process aims to achieve high accuracy and specificity in the automated analysis of road surface conditions.

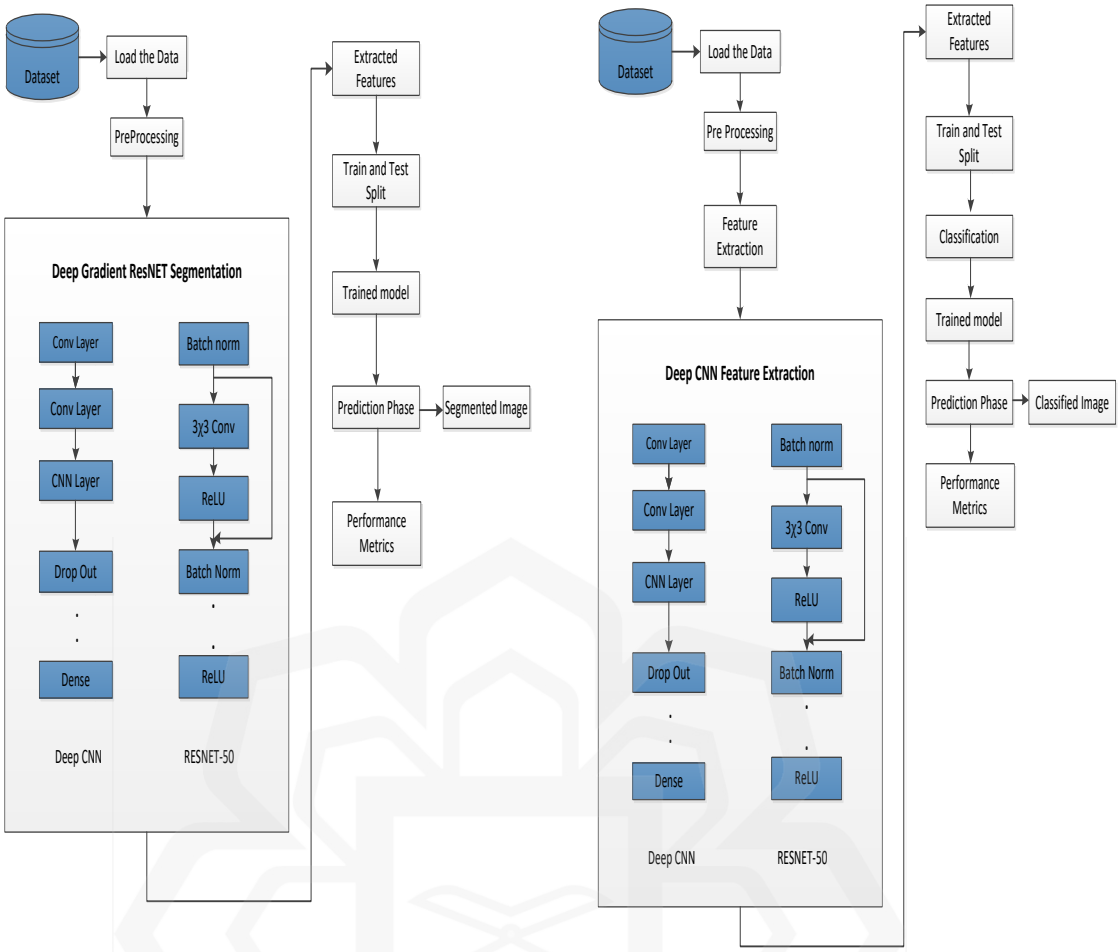



Figure 3.8. Proposed Research Methodology

The initial phase of the methodology commences with the comprehensive dataset acquisition, succeeded by a rigorous preprocessing regimen to condition the data for optimal neural network performance. The preprocessed data is first channeled into the segmentation phase, which is powered by a Deep Gradient ResNet architecture. This phase is meticulously constructed with conv_blocks comprising convolutional layers, ReLU activations, and batch normalization, tailored for the nuanced task of extracting detailed spatial features of road cracks.

Central to this phase is the innovative use of an Attention mechanism that intelligently enhances the model's focus on pertinent features within the residual connections, thereby refining the segmentation process. The output of this stage is a

segmented image that delineates the cracks, providing a clear target for the subsequent classification phase.

Following segmentation, the classification pathway leverages the robust foundation of the ResNet-50 architecture. It employs a sequence of convolutional layers, batch normalization, and ReLU activation functions, interspersed strategically with dropout layers to prevent overfitting. This deep CNN structure is instrumental in distilling the segmented features into a form amenable to classification.

The concluding segment of this pipeline is the application of the trained model to predict outcomes, which are then meticulously evaluated using an array of performance metrics. These metrics not only attest to the model's precision but also provide a quantitative assessment of its performance in the real-world task of crack detection and characterization.

### 3.5.1. Data Acquisition for Enhanced Road Crack Analysis

In the realm of deep learning, the caliber of the dataset is a cornerstone that dictates the success of model training and the veracity of subsequent validations. It is well-established that a dataset's accuracy and heterogeneity are critical to the soundness and generalizability of deep learning models. To this end, the data acquisition segment of our methodology adopts an innovative strategy to encapsulate the complexities of road surfaces within Selangor and Kuala Lumpur.

Capitalizing on the advanced imaging capabilities of a GoPro Hero 8 camera, our team devised a custom mounting apparatus, specifically tailored and attached to a vehicle designated for inspection. This bespoke assembly, a first-of-its-kind for this type of research, allowed for the systematic capture of high-fidelity road surface data. The selection of the GoPro Hero 8 was strategic, owing to its high resolution and dynamic range, ensuring the clarity and detail of road anomalies were preserved in the collected footage.

Diverging from conventional methods, the data collection process was engineered to emulate authentic driving conditions meticulously. The camera was strategically positioned at a height and angle to span a 3.1-meter breadth of the road, which mirrors the average lane width encountered in the targeted geographical locales. This precise configuration was instrumental in acquiring data that not only embodies realistic road scenarios but also enriches the dataset with a level of detail that is often absent in comparative studies.

This alignment is more than a technicality; it is a thoughtful consideration that bridges the gap between simulated test environments and the variable nature of real-world road conditions. By adopting such a realistic capture strategy, the dataset elevates the potential for the developed model to understand and interpret typical road scenarios more accurately. The implications of this approach are significant, paving the way for groundbreaking enhancements in the field of automated road analysis.

In addition to the original data collected using the GoPro Hero 8 camera, this research also utilized the CRACK500 dataset to supplement and diversify our training and validation data. The CRACK500 is a well-regarded, publicly available dataset that consists of 500 images of road cracks, annotated with precise ground truth labels for crack detection and segmentation tasks. These images encompass a variety of road types, conditions, and lighting scenarios, providing a broad spectrum of real-world data that is invaluable for training deep learning models.

The inclusion of the CRACK500 dataset serves several purposes. First, it enhances the volume of data available for the model to learn from, which is crucial for deep learning algorithms to perform well. More data leads to better model generalization, allowing the algorithm to learn the intricacies of road crack patterns and variations more effectively. Second, it introduces a wider range of crack types and road conditions to the model, ensuring that it does not overfit the specific characteristics of the roads in Selangor and Kuala Lumpur alone.

By combining the CRACK500 dataset with the data collected from local roads, the model gains exposure to a more comprehensive array of crack instances, leading to improved robustness and accuracy in crack detection and characterization across

various environments. This blended dataset approach underpins the model's ability to perform consistently well in a multitude of real-world scenarios, marking a significant stride in the advancement of automated road analysis technologies. Representational pictures from the combined dataset are shown in Figure 3.9.

Figure 3.9. Representational Images from our dataset

### 3.5.2. Train-Test Split

The train-test split is a common practice in machine learning that divides the dataset into two parts: one for training the model and one for testing its performance. In this case, the dataset has been split in a 90:10 ratio, meaning that 90% of the images are used for training and 10% for testing.

Using 90% of the images for training allows the model to learn from a vast majority of the data, which can lead to a better understanding of the patterns and features associated with road cracks. It gives the model ample examples to learn from, which is crucial for deep learning models that require a lot of data to generalize well.

The remaining 10% of the images are set aside to test the model. This test set is a separate, untouched portion of data that the model has never seen before. It acts as a new set of examples to evaluate how well the model can apply what it has learned to

new, unseen data. This can give us a good indication of how the model will perform in real-world scenarios, where it will encounter data, it has never seen during training.

Splitting the dataset into 90% for training and 10% for testing can indeed raise concerns about potential overfitting. However, this approach is justified for several reasons, especially in the context of training a model from scratch as in this case on real-world data for pavement crack detection and characterization:

1. Large Training Data: Using 90% of the data for training ensures that the model is exposed to a wide variety of crack types and conditions, which is crucial for learning the complex patterns inherent in real-world pavement cracks.
2. Real-World Variability: Real-world data is often noisy and varied. A larger training set helps the model generalize better by learning from diverse examples, reducing the risk of overfitting compared to synthetic or less varied data.
3. Model Robustness: For complex tasks like pavement crack detection, having more training data allows the model to develop a more robust understanding of the features that distinguish different crack types, leading to better performance even on unseen data. Table 3.10 shows the split of images for training and testing.

Table 3.10. Train Test Split

| **Total Images** | **Training Set Images** | **Testing Set Images** |
|---|---|---|
| 3699 | 3329 | 370 |

### 3.5.3. Data Preprocessing

In training deep learning models for road crack detection, data preprocessing is a pivotal step that ensures models receive the best possible input to learn from. This stage includes cleaning, normalizing, and transforming raw data to eliminate biases

and irregularities, which is fundamental for the effective training of algorithms. Proper preprocessing not only hastens the learning process but also amplifies the models' ability to generalize and interpret data accurately.

Our research employed a systematic and advanced approach to preprocessing. We began by extracting individual frames from videos captured by a camera mounted on a vehicle, thus transforming continuous visual information into discrete, analyzable images. These images were then resized to a uniform resolution of 256x256 pixels, which not only provided consistency but also optimized them for computational efficiency in the subsequent processing stages.

To bolster the diversity of our dataset and enhance the robustness of our model, we introduced variations through horizontal and vertical flips, each with a 50% chance of being applied to any given image. This approach simulated a broader range of perspectives and real-world road scenarios. In addition, we carefully adjusted the brightness and exposure of the images, modifying the brightness by up to ±40% and the exposure by up to ±16%, to replicate the various lighting conditions a vehicle might encounter.

The CRACK500 dataset, an established collection of road crack images, underwent the same meticulous preprocessing. It was important that the CRACK500 images were treated equally to ensure uniformity across the board. These images, like our own, were resized, flipped, and had their lighting conditions adjusted to ensure they were representative of the diverse environments one might encounter.

Furthermore, for the multi-class segmentation aspect of our project, label encoding was applied to both our collected images and the CRACK500 images. This labeling is crucial for training the model to identify and distinguish between different types of road damage.

By subjecting both the locally collected images and the CRACK500 dataset to this rigorous preprocessing regimen, we aimed to cultivate a dataset that was not only extensive but also reflective of the multifaceted conditions of real-world roads. This sets a strong foundation for training a machine learning model with enhanced

accuracy and adaptability, ensuring it's well-equipped to handle the complexities of road crack detection and analysis.

### 3.5.4. Deep Gradiant Resnet Segmentation

The Deep Gradient ResNet Segmentation approach is a sophisticated method in machine learning that combines the strengths of Deep Learning, Gradient-based learning, and Residual Networks (ResNet) for the task of image segmentation. Image segmentation is the process of partitioning an image into multiple segments, or sets of pixels, often with the goal of simplifying or changing the representation of an image into something that is more meaningful and easier to analyze.

Here's a simplified breakdown of the components in Deep Gradient ResNet Segmentation:

1. Deep Learning: This is a class of machine learning algorithms that use multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input. In the context of image segmentation, deep learning can process the input images through these layers to learn the features necessary for accurate segmentation.
2. Gradient-Based Learning: This involves using the gradient of the loss function to update the network parameters in the direction that will minimize the loss. The gradient is a multi-dimensional slope that tells you how to change your parameters to make the model's predictions more accurate.
3. Residual Networks (ResNet): ResNet architecture introduces the concept of a "residual block" which includes skip connections that jump over one or more layers. Traditional deep learning models struggle with the vanishing gradient problem as they get deeper, but ResNet's skip connections allow for the training of very deep networks by enabling the gradient to flow through the network directly.
4. In segmentation, particularly with the Deep Gradient ResNet approach, the aim is to classify each pixel in the image into a category (such as "road

crack" or "not road crack"). The ResNet model would go through layers, learning to identify features of road cracks, and the gradient-based optimization would tweak the model's weights to improve its accuracy over time.

To provide a visual explanation, Figure 3.10 presents a simplified representation of a segmentation task using the Deep Gradient ResNet approach.

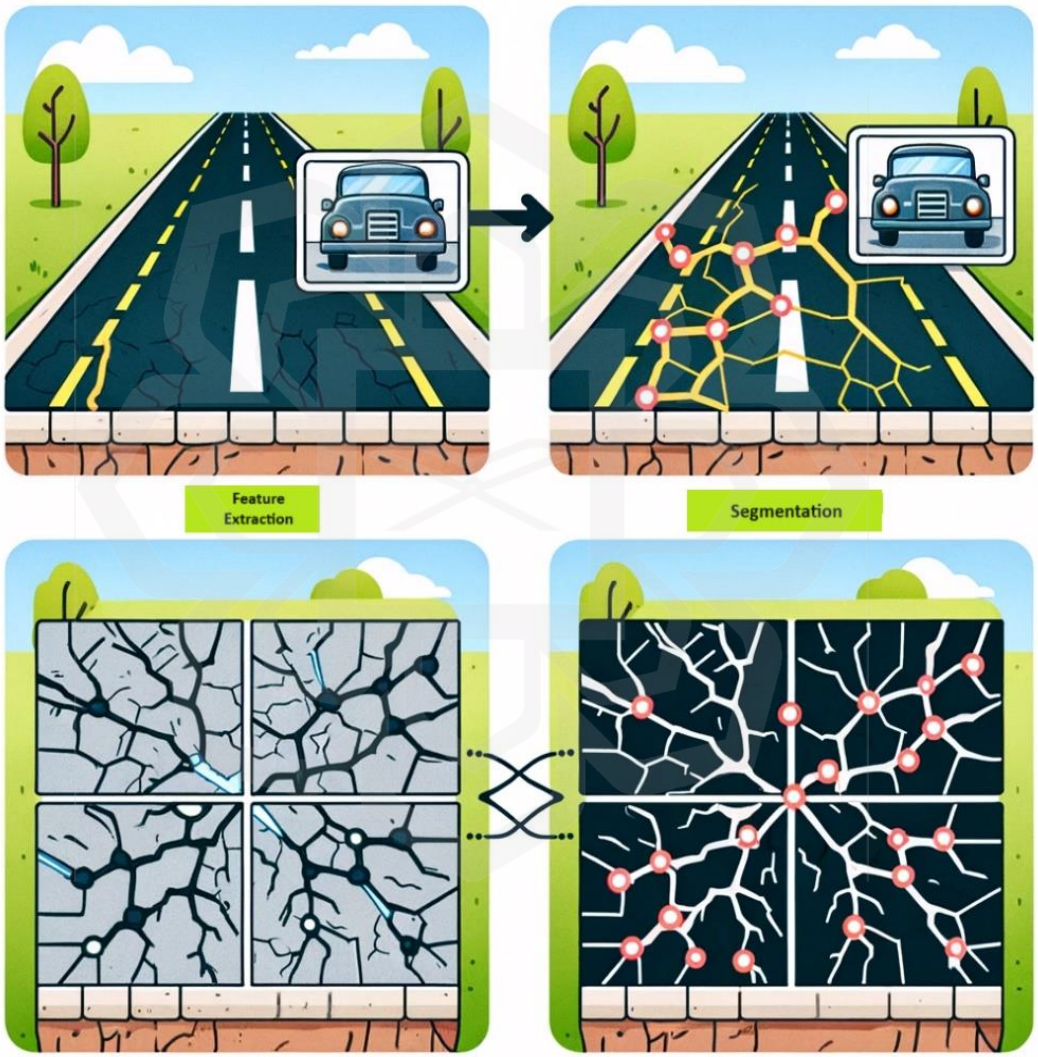



Figure 3.10. Deep Gradient ResNet Approach

- Input Image: An image of a road surface that needs to be analyzed for cracks.

- Feature Extraction: Layers of the ResNet model process the image and identify features relevant to road cracks.
- Segmentation Map: The model produces an image where each pixel's classification corresponds to a category, with road cracks marked distinctly.
- Output Image: A segmented image that highlights the cracks.

The Deep Gradient ResNet approach combines deep learning, gradients, and Residual Networks (ResNet) to perform tasks like image segmentation. The mathematical concepts involved in this approach:

1. **Residual Learning**: ResNet introduces residual learning to ease the training of networks that are substantially deeper than those used previously. The core idea is to fit a residual mapping $F(x)$ rather than the desired underlying mapping $H(x)$. This is done by using shortcut connections (also called skip connections) that skip one or more layers. The original mapping is recast into $F(x) + x$.
2. **Layer Operations**: A typical ResNet layer for an input $x$ will have the following operation as shown in Equation 3.4:

$$x_{l+1} = x_l + F(x_l, W_l) \tag{3.4}$$

   where $x_l$ is the input to the *l*-th layer, $W_l$ are the weights of the *l*-th layer, and $F$ is the residual function (for example, a stack of two 3×3 convolutions).
3. **Batch Normalization**: After each convolutional operation, batch normalization is applied, which normalizes the output of the previous layer by subtracting the batch mean and dividing by the batch standard deviation. Mathematically shown in Equation 3.5:

$$x = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{3.5}$$

where $\mu_B$ is the batch mean, $\sigma_B^2$ is the batch variance, and $\epsilon$ is a small constant for numerical stability.

4. **Activation Functions**: A ReLU (Rectified Linear Unit) activation function is often used, which is as shown in Equation 3.6:

$$ReLU(x) = max(0, x) \tag{3.6}$$

This introduces non-linearity into the model and allows for complex functions to be learned.

5. **Gradient Descent**: The parameters of the network are updated using gradient descent, where the update rule for a weight $W$ at iteration $t$ is given by Equation 3.7:

$$Wt + 1 = Wt - \eta \cdot \nabla WJ(Wt) \tag{3.7}$$

Here, $\eta$ is the learning rate and $\nabla WJ(W)$ is the gradient of the loss function $J$ with respect to the weights $W$.

6. **Loss Function for Segmentation**: For segmentation tasks, a common loss function is the cross-entropy loss, calculated over the spatial dimensions of the output as shown in Equation 3.8.

$$J(W) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{h,w} y_{i,h,w} \log\left(\hat{y}_{i,h,w}(W)\right) \tag{3.8}$$

where $N$ is the number of samples, $y$ is the true label, $\hat{y}$ is the predicted label, and $h, w$ are the spatial dimensions of the image.

7. **Backpropagation**: Backpropagation is used to calculate the gradient of the loss function with respect to the weights. In ResNets, the gradient can flow directly through the skip connections, making it easier to train deeper networks.

### 3.5.5. Development of Deep Gradient ResNet for Pavement Crack Segmentation

In the burgeoning field of image segmentation, the novel segmentation model introduced here stands as a testament to innovative engineering and advanced computational strategies. At its core, the segmentation process is fortified by the conv_block function, a designed sequence of convolutional layers paired with batch normalization and ReLU activation. This foundational structure is pivotal in enabling the model to discern and assimilate complex features from the input images with remarkable efficiency.

The incorporation of batch normalization is a strategic choice, serving dual purposes: it expedites the model's convergence rate and fortifies its capacity to generalize: both essential qualities for the reliability and adaptability of the segmentation task.

A groundbreaking addition to the model is the Augmented SubPixel Shuffling (ASPS) Layer, materialized through the AugmentedSubPixelShuffling class. This innovative layer harnesses the technique of sub-pixel shuffling to spatially upscale the resolution of feature maps, thereby significantly enhancing the model's precision in localizing and segmenting images. In the realm of segmentation, where accuracy is paramount, the ASPS layer emerges as a game-changer.

The model's prowess is further amplified by the proposed Deep Gradient ResNet Architecture, encapsulated within the Deep_Gradient_ResNet class. This architectural marvel intertwines encoder and decoder blocks in an elegant symphony, adeptly encoding and decoding information to capture an array of features spanning local to global scopes.

One of the most salient features of this architecture is the strategic use of skip connections. These connections create conduits for information flow between the encoder and decoder blocks, preserving critical spatial details and facilitating the model's learning from multiple scales. This multi-scale learning is essential for the nuanced understanding necessary for precise segmentation.

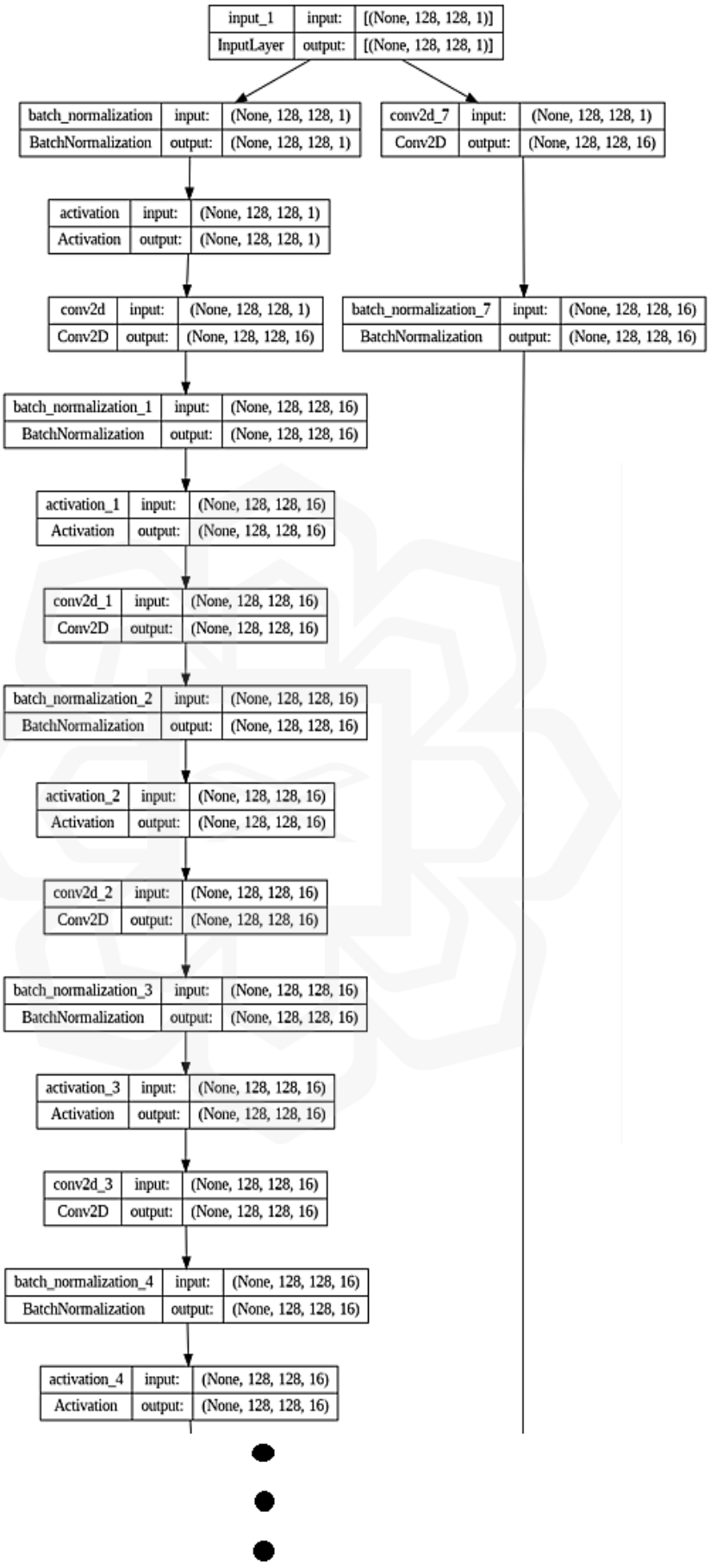
Another significant architectural element is the use of transposed convolutional layers. These layers, known for their upsampling prowess, enable the model to refine the spatial resolution of the feature maps. Learning to increase the granularity of these maps is crucial, as it directly impacts the model's segmentation performance. Figure 3.11 covers all steps in a continuous, structured format with the necessary coding-algorithm syntax, incorporating each stage of the methodological innovation for segmentation model development.

| **Algorithm Code Syntax** |
| --- |
| **Initialize**<br>`input_image = {resolution: 128x128, channels: 3}`<br>`conv_block_config = {filters: 16, kernel_size: 3x3, activation: ReLU}`<br>`deep_resnet_config = {encoder_layers: 5, decoder_layers: 5}`<br>Initialize `Deep_Gradient_ResNet` architecture. |
| **For each convolutional block:**<br>Apply `Conv2D(filters, kernel_size)`<br>Apply `BatchNormalization()`<br>Apply `ReLU activation`<br>Store processed feature maps for skip connections. |
| **Deep Gradient ResNet Architecture:**<br>**For each encoder layer:**<br>`features = extract_features(input_image, layer=encoder_layer)`<br>Store `features` for decoding.<br>**For each decoder layer:**<br>`upsampled_features = TransposedConv2D(features)`<br>Incorporate `skip_connection(encoder_features, decoder_features)` to preserve spatial details. |
| **Augmented SubPixel Shuffling Layer:**<br>`upsampled_maps = AugmentedSubPixelShuffling(feature_maps)`<br>Upsample feature maps using sub-pixel shuffling.<br>`precision_enhanced = improve_precision(upsampled_maps)` for accurate segmentation by improving spatial resolution. |
| **For each skip connection between encoder and decoder layers:** |

| |
|---|
| `skip_connection_data = pass_spatial_information(encoder_features)`<br>Pass `skip_connection_data` to decoder to enhance multi-scale learning and improve segmentation accuracy. |
| **Transposed Convolutional Layers for Upsampling:**<br>**For each decoder block:**<br>`refined_maps = TransposedConv2D(decoder_features)`<br>Increase spatial resolution of `refined_maps` for more precise segmentation. |
| **Model Flexibility and Adaptation**<br>**Configure model flexibility:**<br>`model = adapt_model(segmentation_task)`<br>Adjust for different segmentation tasks.<br>Enable multi-scale feature extraction for enhanced robustness. |
| **Generate output:**<br>`segmented_output = model.predict(input_image)`<br>Evaluate segmentation accuracy using `metrics = {precision, recall, IoU}`<br>`print(metrics)` |

Figure 3.11. Proposed Algorithm for Deep Gradient ResNet-Based Segmentation Model

Flexibility is woven into the very fabric of the model, allowing it to adapt seamlessly to a variety of segmentation features and handle larger features with ease. This adaptability makes the model a robust tool capable of tackling diverse and challenging segmentation tasks. By integrating these advanced components, the segmentation model not only pushes the boundaries of current methodologies but also sets a new benchmark for accuracy and adaptability in image segmentation technologies. The architectural diagram, in Figure 3.12, provides a visual encapsulation of the model's intricate design and sophisticated functionality.


input_1 | input: | [(None, 128, 128, 1)]
InputLayer | output: | [(None, 128, 128, 1)]
batch_normalization | input: | (None, 128, 128, 1)
BatchNormalization | output: | (None, 128, 128, 1)
conv2d_7 | input: | (None, 128, 128, 1)
Conv2D | output: | (None, 128, 128, 16)
activation | input: | (None, 128, 128, 1)
Activation | output: | (None, 128, 128, 1)
conv2d | input: | (None, 128, 128, 1)
Conv2D | output: | (None, 128, 128, 16)
batch_normalization_7 | input: | (None, 128, 128, 16)
BatchNormalization | output: | (None, 128, 128, 16)
batch_normalization_1 | input: | (None, 128, 128, 16)
BatchNormalization | output: | (None, 128, 128, 16)
activation_1 | input: | (None, 128, 128, 16)
Activation | output: | (None, 128, 128, 16)
conv2d_1 | input: | (None, 128, 128, 16)
Conv2D | output: | (None, 128, 128, 16)
batch_normalization_2 | input: | (None, 128, 128, 16)
BatchNormalization | output: | (None, 128, 128, 16)
activation_2 | input: | (None, 128, 128, 16)
Activation | output: | (None, 128, 128, 16)
conv2d_2 | input: | (None, 128, 128, 16)
Conv2D | output: | (None, 128, 128, 16)
batch_normalization_3 | input: | (None, 128, 128, 16)
BatchNormalization | output: | (None, 128, 128, 16)
activation_3 | input: | (None, 128, 128, 16)
Activation | output: | (None, 128, 128, 16)
conv2d_3 | input: | (None, 128, 128, 16)
Conv2D | output: | (None, 128, 128, 16)
batch_normalization_4 | input: | (None, 128, 128, 16)
BatchNormalization | output: | (None, 128, 128, 16)
activation_4 | input: | (None, 128, 128, 16)
Activation | output: | (None, 128, 128, 16)

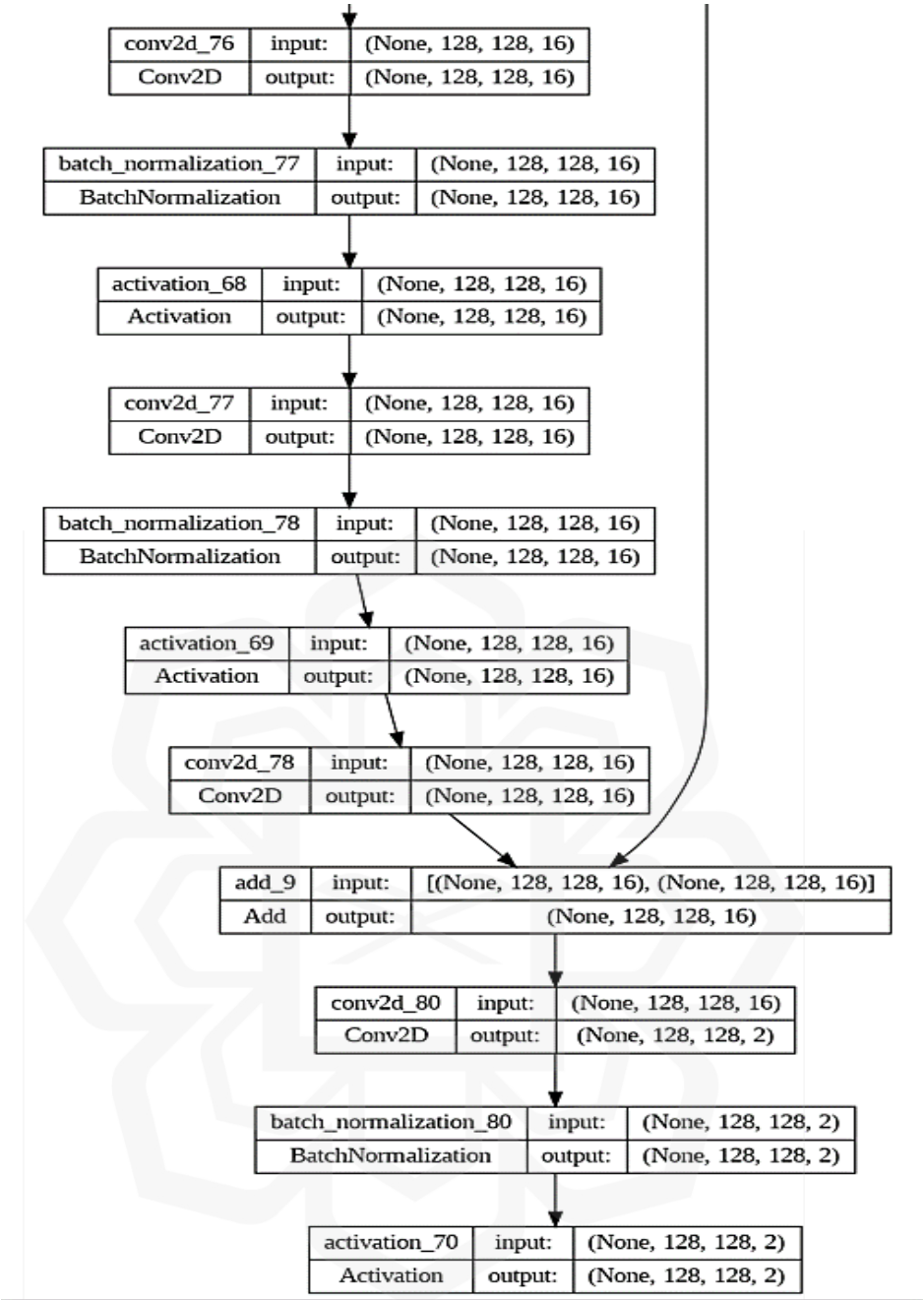



Figure 3.12 Architectural Diagram of segmentation model (initial and final blocks)

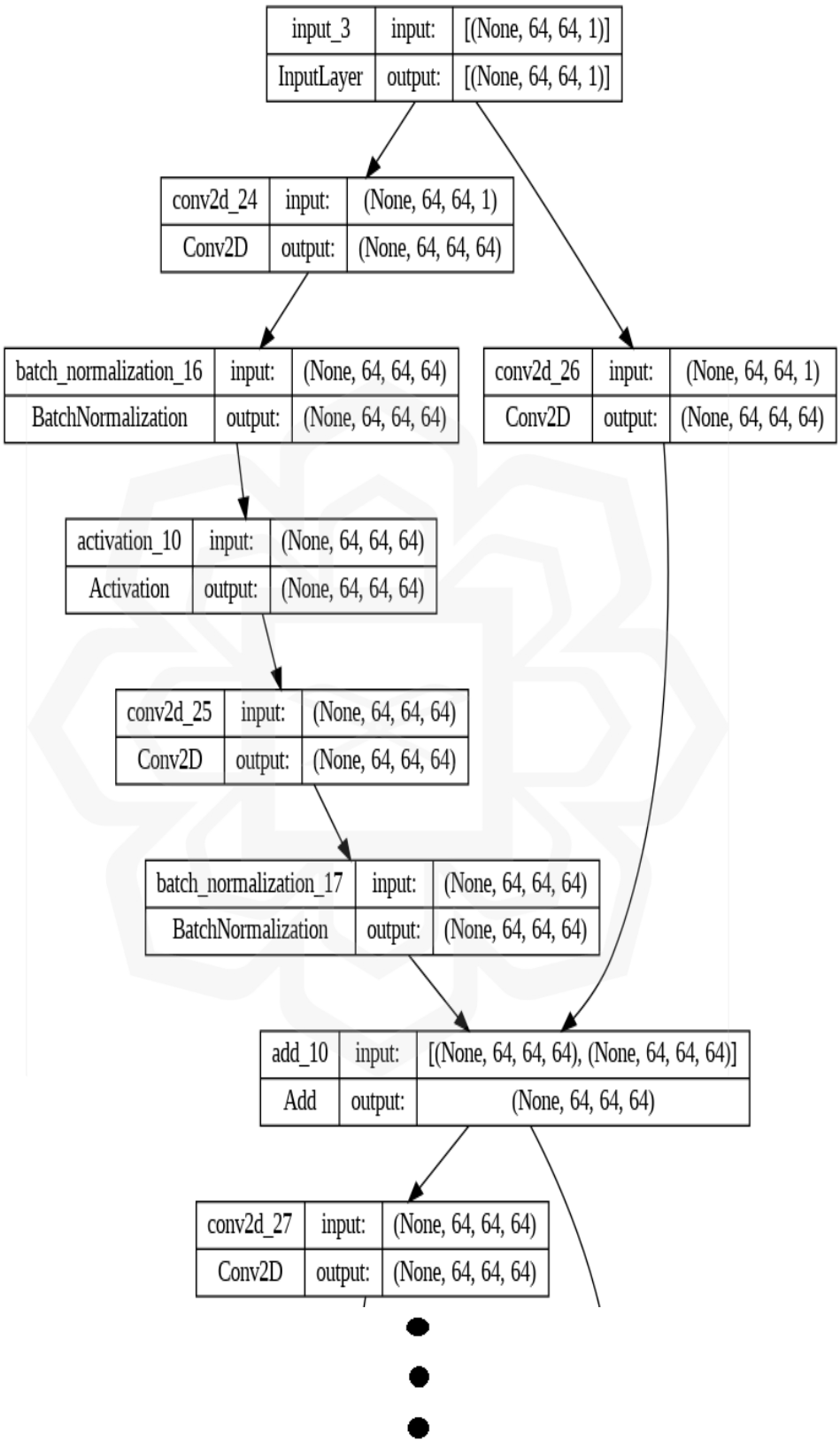
### 3.5.6. Classification Using Resnet and Attention Mechanism

This section introduces several innovative enhancements to the traditional Convolutional Neural Network (CNN) architectures, specifically focusing on the ResNet model as presented in Figure 3.13. These enhancements are designed to

address common challenges in deep learning and improve the model's performance on complex tasks. The architecture of the model is shown in Figure 3.14.

| **Algorithm Code Syntax** |
|---|
| **For each residual block:**<br>`residual_output = x + f(x)`<br>`residual_output = apply_activation(residual_output)`<br>Allows for deeper networks by learning residual mappings instead of direct mappings. |
| **For each convolutional layer:**<br>`conv_output = Conv2D(filters, kernel_size)(input)`<br>`batch_norm_output = BatchNormalization()(conv_output)`<br>`activation_output = ReLU()(batch_norm_output)` |
| **Apply attention mechanism:**<br>`attention_scores = attention_layer(input)`<br>`focused_output = input * attention_scores`<br>Improves feature focusing on important regions, reducing noise from less relevant areas. |
| **After each feature map:**<br>`global_avg_output = GlobalAveragePooling2D()(input)`<br>`global_max_output = GlobalMaxPooling2D()(input)`<br>`combined_output = Add()([global_avg_output, global_max_output])` |
| **After pooling layers:**<br>`flatten_output = Flatten()(combined_output)`<br>`dense_output = Dense(units, activation='ReLU')(flatten_output)`<br>Adds high-level abstraction to enable better predictions. |
| **For each residual connection:**<br>`attention_scores = attention_layer(input)`<br>`residual_output = input * attention_scores + f(input)`<br>Enhances the selective learning of features via attention guidance. |
| **For each residual block in specialized data:**<br>`custom_conv_output = Conv2D(filters, kernel_size, strides=strides)(input)`<br>Allows for adaptation to specific datasets by fine-tuning the architecture. |

Figure 3.13. Proposed Algorithm for CNN-ResNet Classification Model


input_3 | input: | [(None, 64, 64, 1)]
InputLayer | output: | [(None, 64, 64, 1)]
conv2d_24 | input: | (None, 64, 64, 1)
Conv2D | output: | (None, 64, 64, 64)
batch_normalization_16 | input: | (None, 64, 64, 64)
BatchNormalization | output: | (None, 64, 64, 64)
conv2d_26 | input: | (None, 64, 64, 1)
Conv2D | output: | (None, 64, 64, 64)
activation_10 | input: | (None, 64, 64, 64)
Activation | output: | (None, 64, 64, 64)
conv2d_25 | input: | (None, 64, 64, 64)
Conv2D | output: | (None, 64, 64, 64)
batch_normalization_17 | input: | (None, 64, 64, 64)
BatchNormalization | output: | (None, 64, 64, 64)
add_10 | input: | [(None, 64, 64, 64), (None, 64, 64, 64)]
Add | output: | (None, 64, 64, 64)
conv2d_27 | input: | (None, 64, 64, 64)
Conv2D | output: | (None, 64, 64, 64)

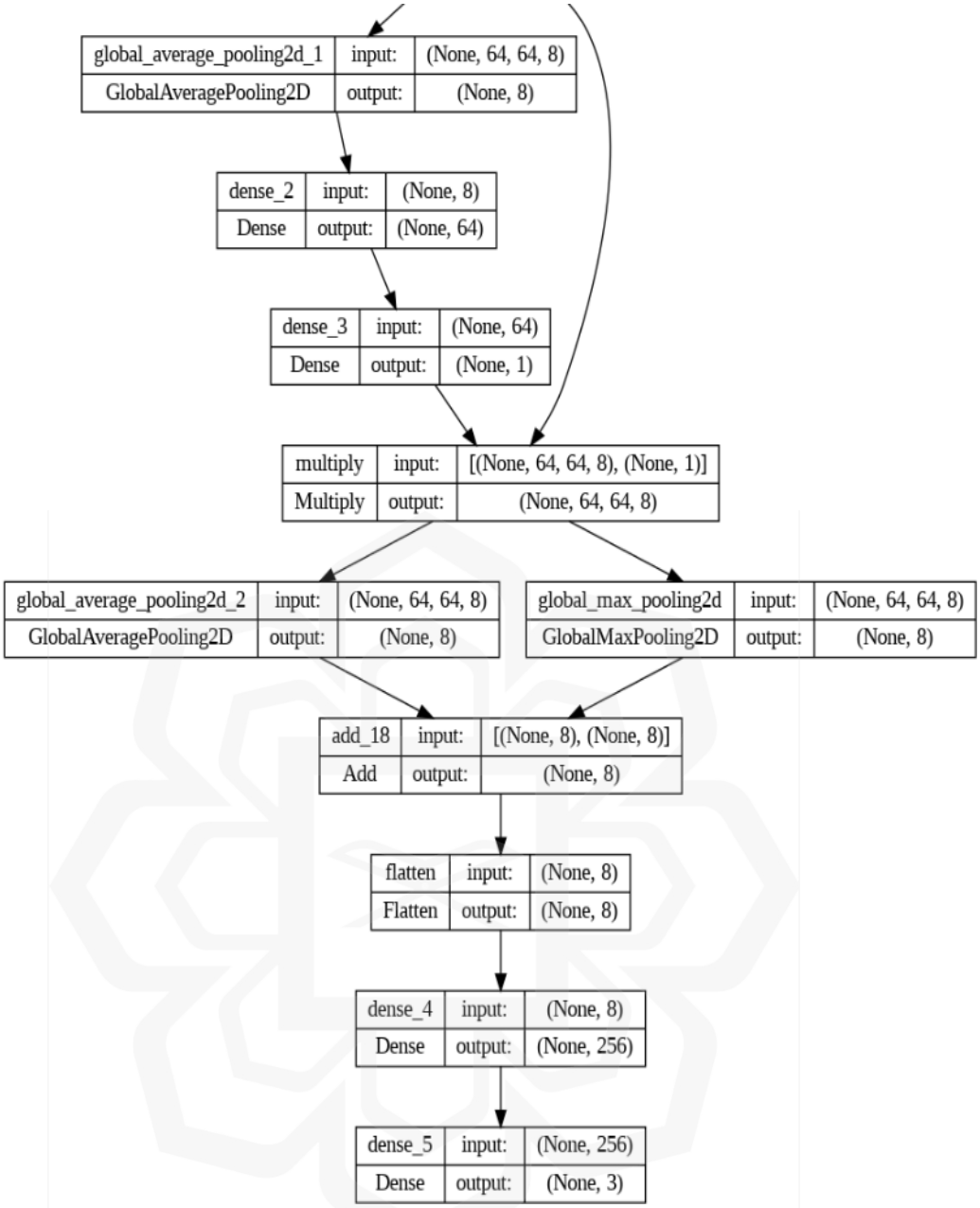



Figure 3.14. Architectural Diagram of classification model (initial and final blocks)

1. Introduction of Residual Blocks: The core of this advancement lies in the incorporation of residual blocks within the network architecture. Residual blocks are a powerful tool that allows the creation of very deep neural networks, which were previously hampered by the vanishing gradient problem. By enabling the network to learn residual mappings instead of direct mappings, these blocks make it easier to train deeper networks, thus enhancing the network's ability to handle more complex tasks.

2. Batch Normalization After Each Convolutional Layer: The implementation of batch normalization after each convolutional layer is another key feature. This technique normalizes the inputs to each layer, reducing the internal covariate shift, which is a common problem in deep learning. By stabilizing the training process, batch normalization facilitates faster convergence, allows for the use of higher learning rates, and ultimately leads to better generalization of the model.
3. Integration of a Proposed Attention Mechanism: The addition of a novel attention mechanism sets this model apart from standard ResNet and CNN models. This mechanism enables the network to selectively concentrate on the most relevant parts of the input data. By focusing on important features and diminishing the influence of less informative regions, the model achieves improved accuracy and robustness, particularly in dealing with complex and large-scale datasets.
4. Combination of Global Average and Max Pooling: The model uniquely combines global average pooling with global max pooling. While average pooling captures the overall trends in the feature maps, max pooling retains the most prominent activations. This dual approach ensures that the model captures both global and local information, enriching the feature representation and enhancing the model's ability to distinguish between different inputs.
5. Dense Layers Post-Pooling: Following the pooling layers, the inclusion of dense (fully connected) layers adds further sophistication to the model. These layers enable the network to learn intricate relationships between the features and the target classes, capturing high-level abstractions in the data, and thereby augmenting the model's predictive capacity.
6. Attention-Guided Residual Connections: A notable innovation is the use of the attention mechanism to guide the residual connections. This involves multiplying the feature maps with attention scores, allowing the model to emphasize or suppress specific features based on their relevance. Such selective processing enhances the overall learning and predictive capabilities of the model.
7. Customized Filter Sizes and Strides: Lastly, the architecture's adaptability and flexibility are boosted through customized filter sizes and strides in

the residual blocks. This customization is particularly beneficial for specialized datasets, like those involving road crack detection, as it allows the model to be fine-tuned for specific types of data.

### 3.5.7. Crack Size Assessment

For the measurement of crack sizes, the methodology incorporated a blend of advanced image processing techniques. The process is methodically structured to accurately quantify crack dimensions in images. Here's an overview of how this is achieved:

1. **Image Preprocessing and Classification**:
   - **Reading and Resizing**: The image (**image_1**) is first read in grayscale using **cv2.imread**. It's then resized to a specific size (**SIZE**).
   - **Binarization**: The resized image is binarized by setting all non-zero pixels to 255. This step highlights the crack areas in the image.
   - **Normalization**: The binary image is normalized to have pixel values between 0 and 1.
   - **Reshaping and Expanding Dimensions**: The image is reshaped and its dimensions are expanded to match the input shape required by the machine learning model.
   - **Model Prediction**: The preprocessed image is fed into a trained model (**model.predict**) to classify the type of crack (e.g., "Alligator", "Longitude", "Transverse").
2. **Crack Analysis Based on Classification**:

   Depending on the classification result (**max_index_1**), different processing steps are taken:
   - **Alligator Cracks**: The image is processed to calculate the area of the crack. The binary format of the segmented image is created, and the area is calculated in pixels and then converted to square meters using a predefined conversion factor.

- **Longitudinal and Transverse Cracks**: For these types of cracks, the length of the crack is calculated in meters using a conversion function (**convert_to_meters**).

3. **Mathematical Calculations**:
   - **Area Calculation**: The area of the crack is computed by summing the pixels in the binary image representing the crack. This value is then multiplied by a conversion factor to get the area in square meters.
   - **Length Calculation**: The length of the crack is determined by analyzing the contours of the crack. The dimensions of the bounding rectangle around the crack are converted from pixels to meters using a conversion factor.
4. **Overlaying Results on Images**:
   The calculated crack size (area or length) and the pixel measurements are then overlayed onto the original images using **cv2.putText**.
   Two images are created, one showing the measurements in meters and the other in pixels.
5. **Blending and Displaying Results**:
   The images with the overlaid measurements are blended with the original image to create a composite image that visually represents the crack measurements. The final blended images are displayed using **matplotlib.pyplot**, showing the results of the crack size analysis.

This process efficiently combines computer vision techniques and machine learning to analyze and quantify crack sizes in images, providing both pixel-based and real-world measurements. The mathematics involved includes pixel count summation for area calculation, contour analysis for length measurement, and the conversion of these measurements from pixels to real-world units (meters).

### 3.5.8. Validation of Crack Length Measurement

To validate the accuracy of crack length measurements obtained through computer vision techniques, a systematic approach was adopted involving both manual

measurement and image processing using a camera positioned at various heights. The following steps outline the methodology in detail:

1. **Image Capture**: Images of marked points on the floor (labeled A, B, C, D, and E) were captured from three different camera heights: 1.57m, 1.60m, and 1.63m as shown in Figure 3.15.


Figure 3.15. Points Under Consideration for Measurement Validation

2. **Manual Measurement**: The actual distances between the marked points were measured manually using a tape measure. The measured lengths were: 0.25m (A-B), 0.50m (A-C), 1m (A-D), and 1.50m (A-E).
3. **Computer Vision Measurement**: Using the captured images, a measurement algorithm was applied to determine the distances between the marked points. This algorithm relied on pixel measurements within the images to calculate the lengths. The detected lengths were then recorded for each camera height.
4. **Comparison and Analysis**: The detected lengths from the image processing system were compared against the manually measured lengths for each camera height. The variations between the detected and actual lengths were calculated to evaluate the accuracy and reliability of the computer vision system.

### 3.5.9. Image Concatenation for Crack Analysis

In this methodology, we concatenate multiple images to create a single frame that can be used for crack analysis. The process involves loading, resizing, concatenating, and visualizing the images. Here's a detailed explanation of the steps involved:

1. Load the Images: First, we load the images that need to be concatenated. These images are read in grayscale mode using OpenCV.
2. Resize the Images: To ensure all images have the same height, we resize them. This step is optional but helps in maintaining uniformity when concatenating the images horizontally or vertically. We determine the minimum height or width among the images and resize each image accordingly, maintaining the aspect ratio.
3. Concatenate the Images: The images are concatenated either horizontally or vertically, depending on the orientation of the cracks. This is achieved by using the np.concatenate function with axis=1 for horizontal concatenation or axis=0 for vertical concatenation. The concatenated image now combines the content of all original images into one frame, providing a continuous view of the cracks.
4. Resize the Concatenated Image: The concatenated image is resized to a predefined size for further processing. This resizing ensures that the image meets the input size requirements of the model, which is crucial for accurate predictions.
5. Model Prediction: The concatenated image is expanded to match the model input shape and is passed to the model for prediction. The model's output is then processed to create a segmented image. The segmentation highlights the crack areas, converting them into a format suitable for analysis.
6. Visualization: Finally, the original concatenated image and the predicted segmented image are visualized side-by-side using Matplotlib. This step provides a clear comparison between the input image and the model's segmentation output. By placing the images side-by-side, we can visually inspect the model's performance and the crack detection accuracy.

## 3.6. SUMMARY

In this chapter we delve into the intricacies of our approach to crack detection and characterization. The methodology's strength lies in its multi-faceted approach, where each model serves a unique purpose and complements the others, thereby providing a comprehensive analysis of road cracks. This diversity in modeling is essential to capture the full spectrum of complexities associated with road crack detection and characterization.

The first model is a customized version of YOLOv7, adapted for the specific task of detecting and categorizing pavement cracks. This model processes a combination of specially gathered road data and publicly available road crack images through innovative preprocessing stages, including image extraction, labeling, augmentation, and resizing. The performance of this model is meticulously evaluated, demonstrating its effectiveness in the context of road crack detection which is presented in Chapter 4.

The second model leverages the advanced capabilities of YOLOv8 X algorithms. This model is fine-tuned for high-accuracy object detection, segmentation, and classification, optimizing its performance within the constraints of our study. The model's unique alterations and optimizations underscore its efficiency in processing and analyzing road crack data.

The third model represents a hybrid approach, ingeniously combining the sophisticated pattern recognition capabilities of deep learning with the reliable and comprehensible nature of conventional algorithms. This fusion results in a potent synergy, enhancing the model's overall effectiveness and robustness in crack detection and characterization.

Data acquisition for this research involved the RDD2022 online dataset, Crack 500, which provides a rich collection of road photographs from diverse geographical locations. This dataset is complemented by custom data captured using a GoPro Hero 8 camera mounted on an inspection vehicle. The data collection process was

meticulously designed, with specific attention to the camera's calibration to accurately represent real-world road conditions.

The data preprocessing phase is comprehensive, involving steps like frame extraction, manual labeling using Roboflow, data augmentation, and resizing. These steps ensure that the data is optimally prepared for processing by the deep learning models. In addition, the methodology extends to crack size assessment. This process incorporates advanced image processing techniques to accurately quantify the dimensions of detected cracks. Depending on the type of crack detected, different methods are employed for measurement, including area and length calculations. The measurements are overlayed on the original images and then blended to create a composite representation. This approach not only highlights the crack areas but also provides precise measurements in both pixels and real-world units.

In conclusion, the methodology outlined in this chapter demonstrates a comprehensive and innovative approach to road crack detection and characterization. The combination of advanced deep learning models and meticulous data processing techniques ensures that the research effectively addresses the challenges in automated crack analysis, paving the way for significant advancements in this field which satisfies the Objective 2 and Objective 3 of this research.

# CHAPTER FOUR

# RESULTS AND DISCUSSION

## 4.1. INTRODUCTION

This chapter delves into the analytical results derived from the ensemble of models detailed in Chapter Three. We will present a detailed evaluation of the outputs from each model, emphasizing their precision and effectiveness in the nuanced task of pavement crack detection and characterization. Our discussion will extend to a systematic benchmarking of these results against established standards, offering a clear depiction of their performance within the broader context of current technological capabilities.

Central to this chapter is the examination of the outcomes from the custom-tailored YOLOv7 model, the precision oriented YOLOv8 X model, and the integrative hybrid model. Each model's results will be meticulously dissected to assess their accuracy, efficiency, and contribution to advancing the automated analysis of pavement cracks. Furthermore, we will scrutinize the calibration setup of our data collection process, presenting how this meticulous design contributed to the accuracy of our models. The calibration's impact on data quality and model training will be highlighted, providing a comprehensive picture of its role in enhancing the precision of crack detection and size quantification.

The chapter will also unfold the findings from the crack size assessment techniques, which have been applied to the datasets meticulously gathered and processed for this research. We will explore the translation of pixel-level detection into meaningful real-world measurements, underpinning the practical significance of our models' capabilities.

## 4.2. EXPERMENTAL SETUP

The experimental setup for this research was carefully structured to ensure an optimal environment for developing and testing the proposed models for pavement crack detection and characterization. The core of our computational resources was an HP Pavilion 15-bc408tx laptop, equipped with robust hardware to handle the demands of deep learning algorithms. The specifics of the computer configuration are as follows:

1. Central Processing Unit (CPU): Intel Core i7-8750H (8th Gen), a high-performance processor capable of executing complex computations rapidly, which is crucial for processing large datasets and running intensive learning algorithms.
2. Random Access Memory (RAM): 16 GB DDR4, providing sufficient memory to manage the data-intensive tasks associated with training deep learning models.
3. Hard Disk Drive (HDD): A spacious 1TB drive, allowing for the storage of extensive datasets, including the RDD2022 and Crack 500 datasets, and the custom data captured for this study.
4. Graphics Processing Unit (GPU): An NVIDIA GeForce GTX 1050 with 4GB of graphics memory, which was used to accelerate the processing of machine learning tasks locally.

In addition to the local computational resources, the research extended to utilize a high-end remote GPU, the NVIDIA GeForce RTX 4080. This powerful GPU significantly enhanced the training speed and efficiency of our models, enabling us to handle more complex computations and larger datasets without the limitations of local hardware resources.

Throughout the course of this study, various Integrated Development Environments (IDEs) were employed to optimize the coding and development process. Anaconda was used as the primary platform due to its streamlined package management and deployment capabilities, which are particularly beneficial for managing multiple deep learning libraries. Google Colab was also utilized, especially for its provision of a powerful cloud-based environment that facilitated access to high-

performance GPUs, which was particularly useful for training the models extensively without the constraints of local hardware.

The software environment for the YOLO models was based on PyTorch, a deep learning library known for its flexibility, ease of use, and native support for GPU acceleration. For the hybrid model, TensorFlow was the framework of choice, offering comprehensive tools and libraries for developing and training machine learning models. Table 4.1 encapsulates the details of the experimental setup used in this research.

Table 4.1. Experimental Setup Specifications

| **Component** | **Specification** |
|---|---|
| System Unit | HP Pavilion 15-bc408tx |
| Processor | Intel Core i7-8750H, 8th Generation |
| Memory | 8 GB, DDR4 Standard |
| Storage | 1 TB Hard Disk Drive |
| Onboard Graphics Unit | NVIDIA GeForce GTX 1050 with 4GB VRAM |
| Remote Graphics Unit | NVIDIA GeForce RTX 4080 |
| Development Platforms | Anaconda, Google Colab |
| Machine Learning Frameworks | PyTorch (for YOLO models), TensorFlow (for Hybrid models) |
| System Unit | HP Pavilion 15-bc408tx |

This setup formed the backbone of the computational platform, empowering the research with the necessary tools to carry out the intensive work of training, testing, and validating the proposed models for effective crack detection and characterization.

## 4.3. CALIBRATION AND DATA ACQUISITION EFFICACY IN ROAD SURFACE IMAGING

In this section, we explore the empirical results stemming from the construction and calibration of a tailored data collection setup, designed to capture high-resolution imagery of road surfaces for the purpose of crack detection. This setup plays a pivotal role in the data acquisition phase of our research, where precision in capturing the full width of the road surface is critical for subsequent analysis.

### 4.3.1. Camera Configuration and Calibration

The core of our data collection apparatus is the GoPro Hero 8 camera, chosen for its high-resolution capabilities and robust performance in varying environmental conditions. The camera specifications are essential to understand the context of the calibration and the quality of data collected as shown in Table 4.2.

Table 4.2. Camera Specifications for Data Collection

| Attribute | Details |
|---|---|
| Camera Model | GoPro Hero 8 |
| Mounting Hardware | GoPro Rod Mount |
| Operating Mode | Custom Video Setting |
| Image Resolution | 1080p |
| Frame Rate Options | 24 fps, 60 fps |
| Lens Type | Linear Field of View |
| Video Bit Rate | Standard Quality (45 Mbps) |
| Minimum ISO Setting | 100 |
| Maximum ISO Setting | 1600 |

This camera was mounted onto a custom setup designed to ensure a consistent 3.1m coverage of the road surface—a critical dimension for the accurate representation of road lane width in our imagery.

### 4.3.2. Lab and Field Calibration Results

The height calibration tests, both in the lab and field environments, were conducted to ascertain the optimal camera height needed to achieve the desired road surface width coverage. The results are as shown in Table 4.3.

Table 4.3. Calibration Results in Lab and Field Settings

| **Setup (Lab)** | **Width** | **Camera angle with respect to ground** |
|---|---|---|
| 1 | 3.1 m | 90° |
| 2 | 3.1 m | 90° |
| **Setup (Field)** | **Width** | **Camera angle with respect to ground** |
| 1 | 3.1 m | 90° ± 35° |
| 2 | 3.1 m | 90° |

These calibration exercises were essential to verify that the camera setup would be suitable for the real-world conditions encountered during the data collection phase.

## 4.4. DATA COLLECTION SETUP

The data collection setup, as visualized in Figure 4.1, was crafted to ensure that the camera's field of view perfectly matched the road width of 3.1m, allowing for comprehensive data collection. The calibration process was not only confined to the lab but was also extended to the field, ensuring that the setup was tested and refined under practical conditions.

Figure 4.1. Setup for Data Acquisition

### 4.4.1. Visual Documentation

To supplement the detailed calibration results and the description of the data collection setup, visual documentation is provided. Figure 4.2 illustrates the lab environment where the calibration was precisely measured, while Figure 4.3 captures the field setup, showcasing the practical application of the camera configuration. These visual aids offer a clear depiction of the experimental arrangements and the fidelity of the calibration process.

Figure 4.2. Lab Calibration Setup

Figure 4.3. Field Data Calibration Setup

By detailing the camera specifications, the calibration results, and the data collection setup, this section provides a transparent and detailed account of the preliminary steps taken to ensure that the data used for model training and analysis was of the highest quality, directly impacting the reliability of our results in detecting and characterizing pavement cracks.

### 4.4.2. Integration And Utilization of External Datasets for Enhanced Crack Analysis

In addition to the custom data collection setup, our research incorporated two external datasets, RDD2022 and Crack500, to augment the diversity and volume of data available for training and validation of our models. These datasets were critical in providing a wide range of crack types and road conditions, which strengthened the models' ability to generalize and perform accurately across various scenarios.

**A) RDD2022 Dataset**

The RDD2022 dataset is an extensive collection of road damage images, which provides a comprehensive view of different road conditions across multiple countries. The dataset's diversity in terms of geographic representation and damage types makes it an invaluable resource for our research as shown on Table 4.4.

Table 4.4. RDD2022 Dataset Overview

| Specification | Detail |
|---|---|
| Source | RDD2022 |
| Total Images (China-Motor bike folder) | 1977 |
| Damage Types | Transverse cracks, Longitudinal cracks, Alligator cracks, Potholes |
| Resolution | Various (High-resolution images) |
| Annotation | Bounding boxes, Damage type labels |
| Usage | Training, Validation |

The RDD2022 dataset was instrumental in training the models to detect and categorize different types of road damage, providing a real-world testing ground to evaluate the models' performance and robustness.

**B) Crack500 Dataset**

The Crack500 dataset, a well-known dataset in the domain of road damage analysis, consists of high-quality images annotated for crack detection and segmentation as shown in Table 4.5. This dataset complements RDD2022 by adding additional variability and aiding in the robustness of the models.

Table 4.5. Crack500 Dataset Overview

| Specification | Detail |
|---|---|
| Source | Crack500 |
| Total Images | 500 |
| Annotation | Pixel-level segmentation masks |
| Resolution | High-resolution images |
| Usage | Training, Validation, Testing |

The Crack500 dataset was particularly useful for training the hybrid model, which relies on pixel-level segmentation to characterize the extent and shape of pavement cracks accurately.

### 4.4.3. Dataset Integration in Model Training

In this research, while the primary data source was our meticulously gathered dataset, to ensure that our models were well-versed with varied real-world scenarios, we supplemented our data with external datasets, specifically RDD2022 and Crack500. These datasets were strategically chosen to enhance the diversity and complexity of the training and validation sets, which is crucial for developing a robust model.

Our custom dataset formed the bulk of the training material, accounting for approximately 70% of the data, reflecting our commitment to creating a model finely tuned to real-world conditions. The remaining 30% of the data was equally divided between RDD2022 and Crack500, each contributing 15% to the overall dataset. This composition was deliberate, ensuring that our models benefit from a broad spectrum of data while still focusing on the high-quality, real-world data collected specifically for this study. Table 4.6 presents a focused distribution, indicating that our custom dataset formed the core of the training and validation data, complemented by RDD2022 and Crack500 datasets to enrich the models' exposure to a variety of crack scenarios and surface conditions.

Table 4.6. Contribution of Datasets to Training and Validation Phases

| **Dataset** | **Contribution to Training Set** | **Contribution to Validation Set** |
|---|---|---|
| Custom Data | 70% | 70% |
| RDD2022 | 15% | 15% |
| Crack500 | 15% | 15% |

## 4.5. RESULTS FROM MODEL 1: CUSTOMIZED YOLOV7

### 4.5.1. Experimental Results from RDD2022

Utilizing the tailored YOLOv7 model on the RDD2022 dataset, and adjusting model settings, resulted in notable precision and recall rates, achieving an aggregate accuracy close to 92%. The confusion matrix and key performance indicators for this particular dataset are depicted in Figure 4.4 and Figure 4.5, respectively.

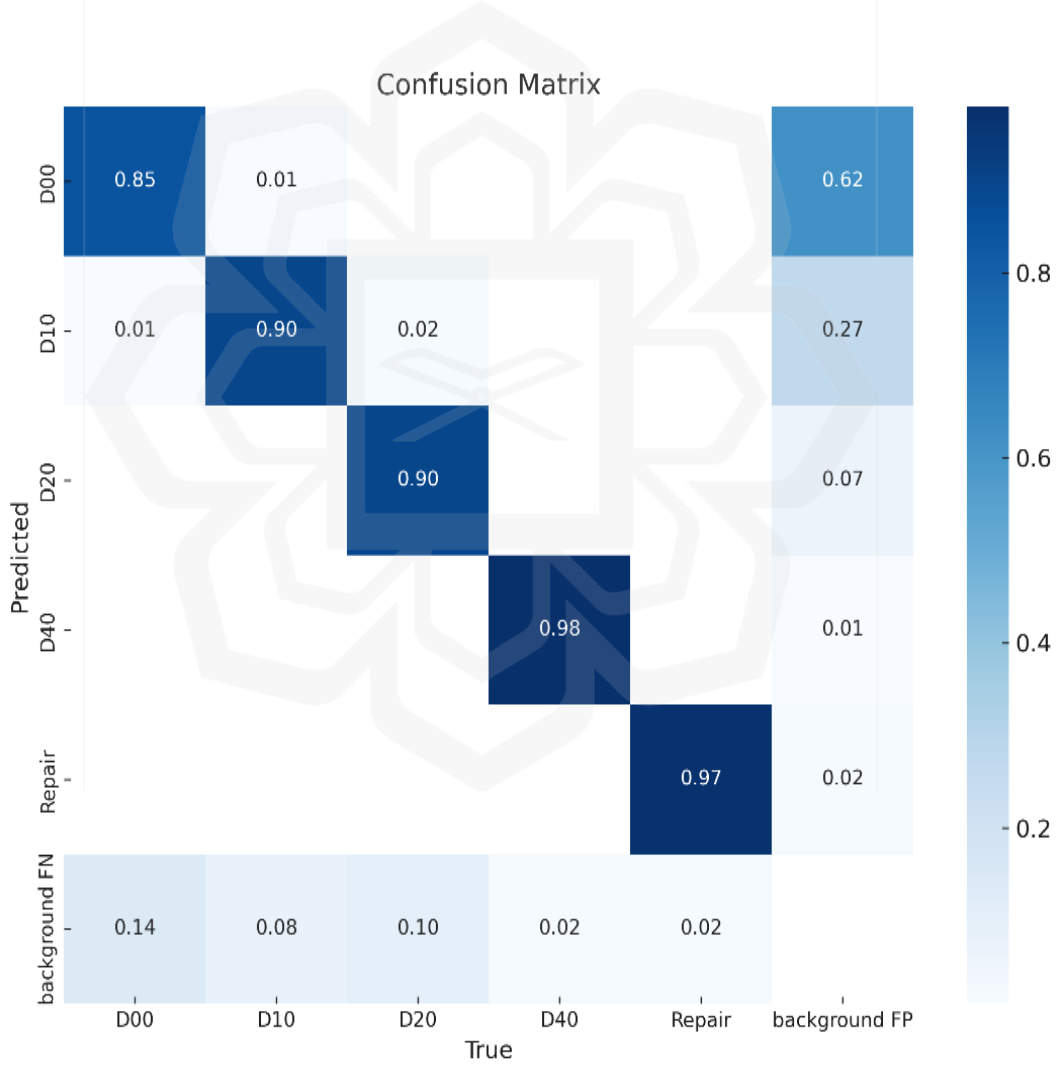



Figure 4.4. Confusion matrix.

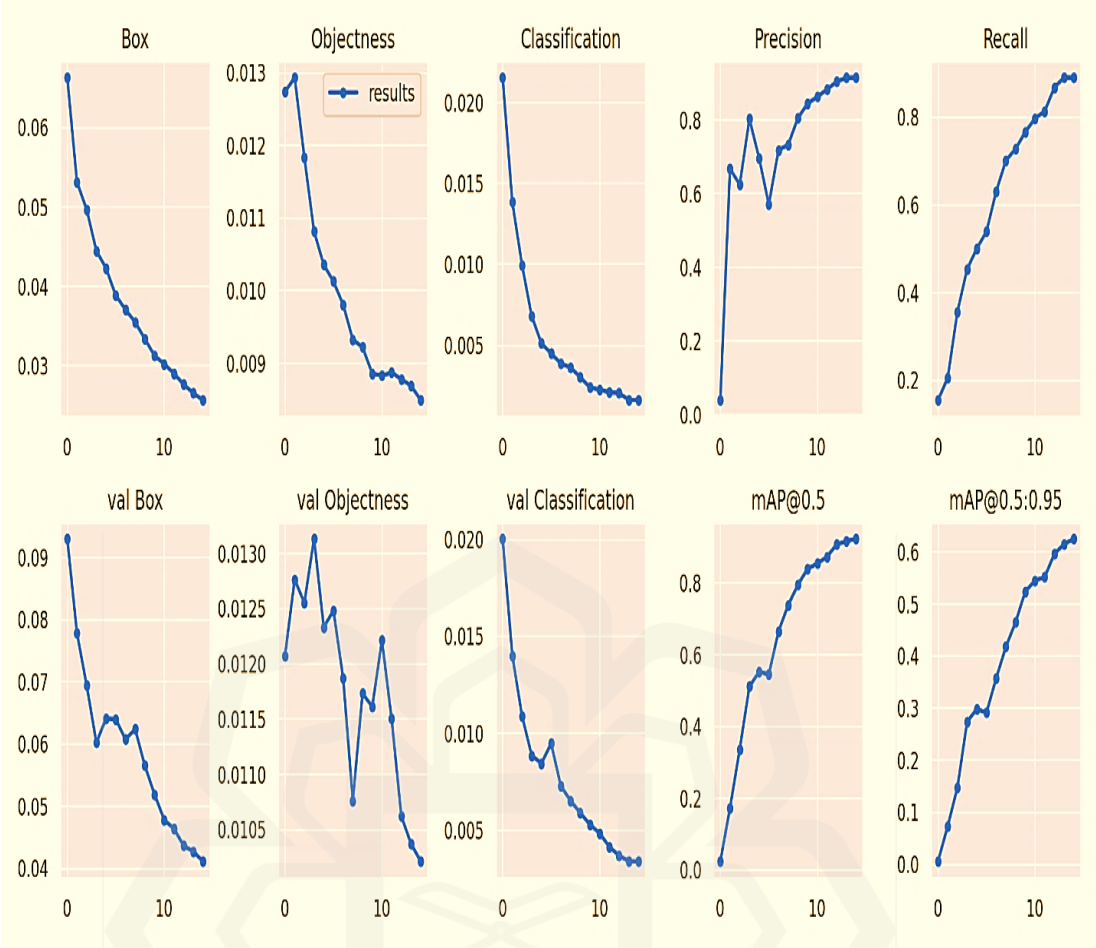



Figure 4.5. Performance metrics.

### 4.5.3. Accuracy Assessment of Crack Types

The accuracy of individual crack types, crucial for understanding the model's performance nuances, is summarized in Table 4.7. The dataset utilized in this research includes a comprehensive array of documented road defects: longitudinal cracks (D00), alligator cracks (D20), transverse cracks (D30), potholes (D40), and areas of repair. The amount of data available for each defect category plays a crucial role in the model's ability to accurately detect and categorize each type of road damage. By integrating state-of-the-art deep learning techniques with advanced image processing methods, the model demonstrated impressive performance.

Table 4.7. Model Performance by Defect Type

| Road Defect | Model Accuracy (%) |
|---|---|
| Alligator Cracks | 90 |
| Longitudinal Cracks | 85 |
| Transverse Cracks | 90 |
| Potholes | 98 |
| Repairs | 97 |
| **Overall Accuracy = 92%** | |

The model's predictions, reflecting the effectiveness of our deep learning algorithms and image-processing techniques, are visually depicted in Figure 4.6.

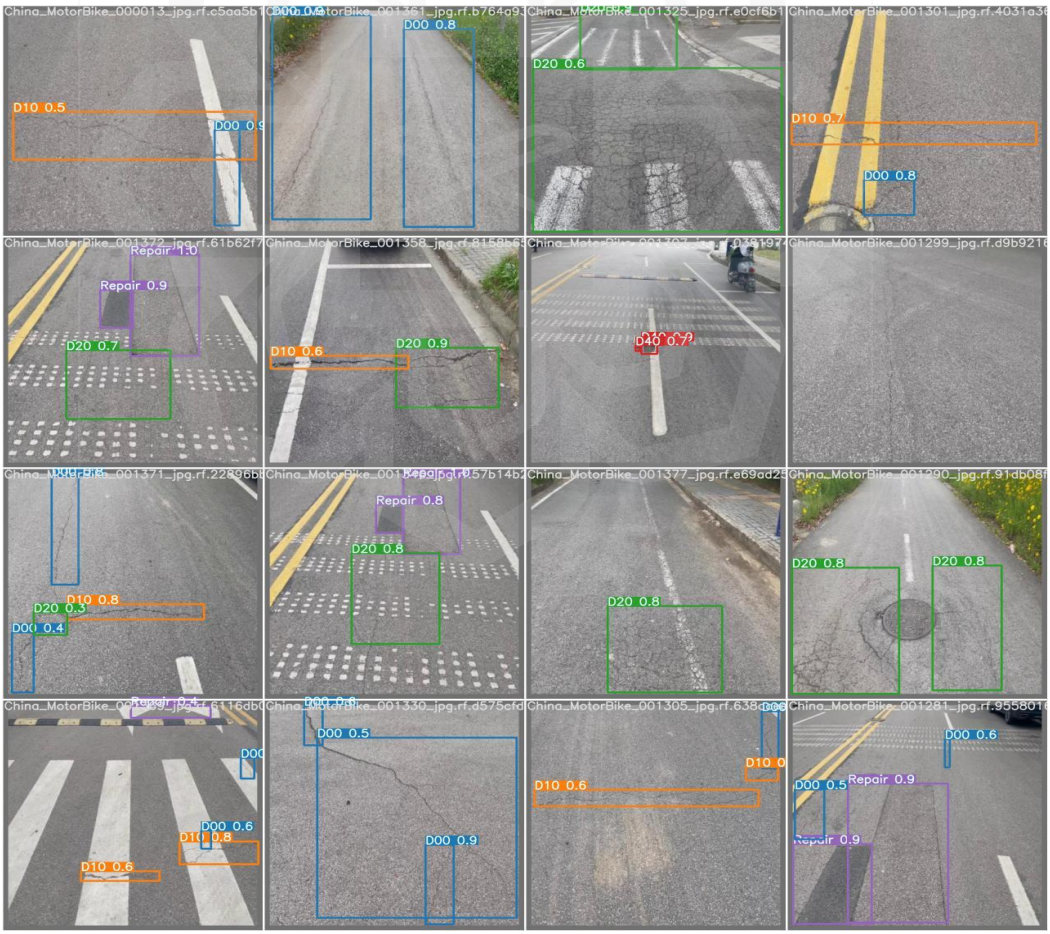



Figure 4.6. Predictions on Testing Data

### 4.5.4. Custom Dataset Results

The custom-configured YOLOv7 model was trained on our exclusive dataset, setting the batch size at 20 and running for 50 epochs. The performance metrics of the model are captured in Figure 4.8. Notably, the model achieved its highest precision, 0.93, at the 35th and 41st epochs, while the peak recall value of 0.9158 was recorded at the 41st epoch. These results underscore the model's proficient ability in both detecting and classifying within real-world scenarios. The accuracy of the model for different types of cracks—namely alligator, longitudinal, transverse, and potholes—is compiled in Table 4.7, leading to an overall accuracy rate of 88%. This achievement signifies a noteworthy advancement over many current models in the field. Figure 4.8 illustrates the model's performance with test data predictions.

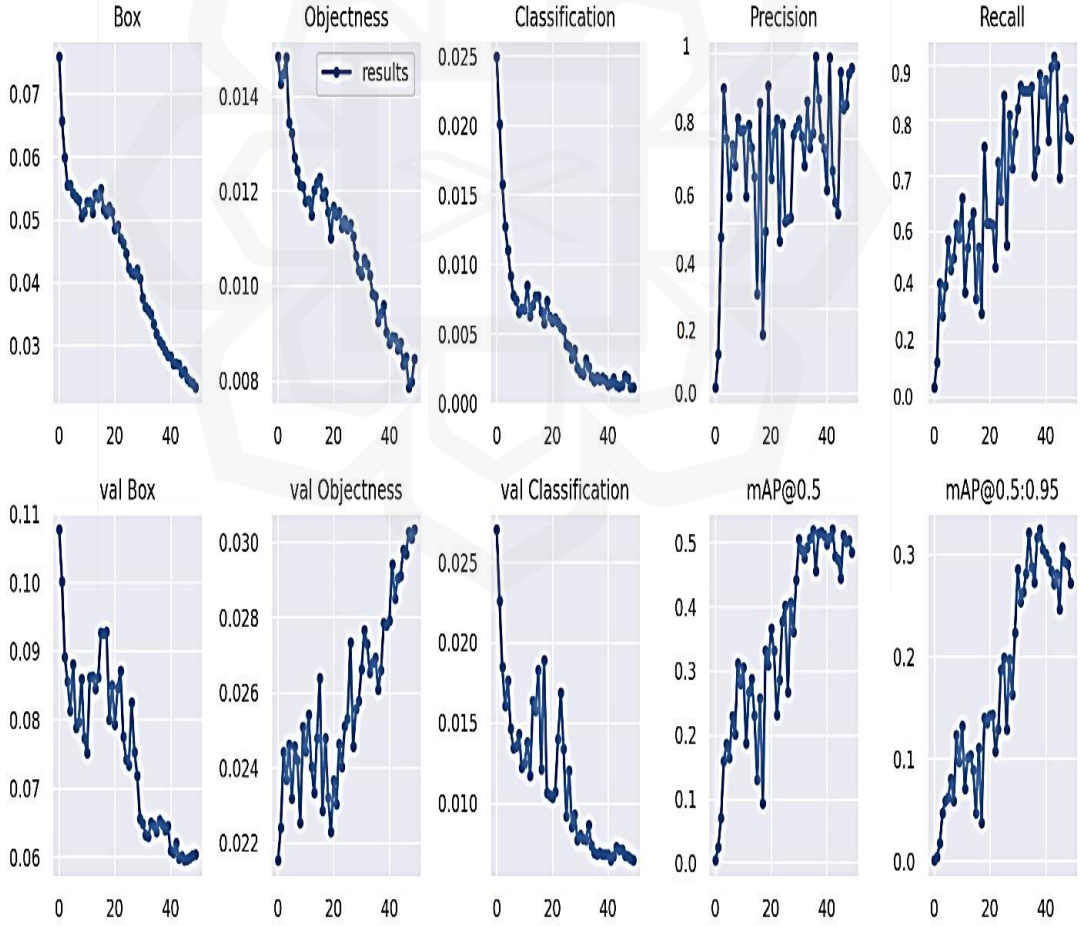



Figure 4.7 Metrics Overview of Model Performance

Table 4.8 Model Performance

| Type of Crack | Accuracy (%) |
|---|---|
| Alligator Cracks | 91 |
| Longitudinal Cracks | 87 |
| Transverse Cracks | 84 |
| Potholes | 90 |
| **Aggregate Accuracy = 88%** | |

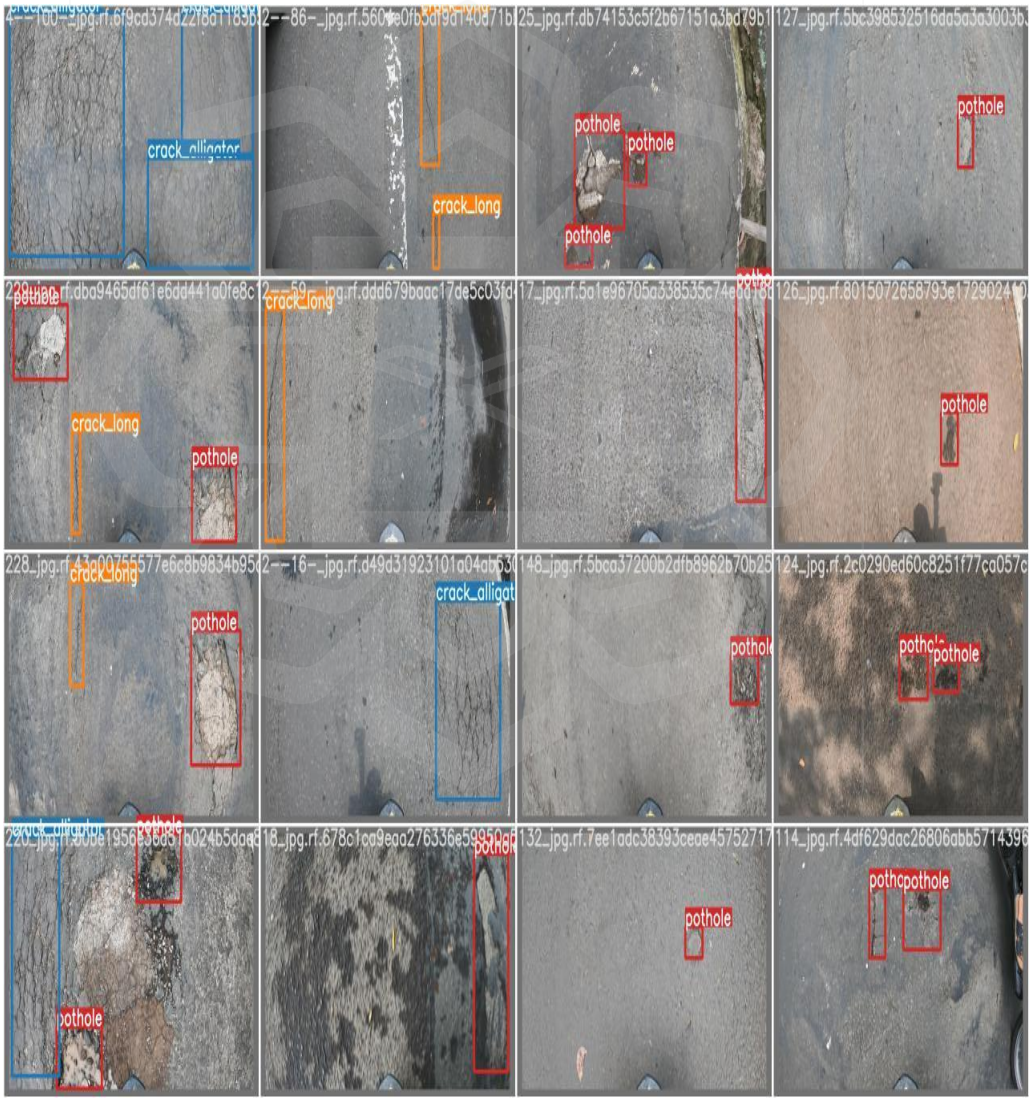



Figure 4.8 Model Predictions on Test Data

### 4.5.5. Benchmarking

This investigation underwent benchmarking against concurrent studies within the same domain. The outcomes derived from this study markedly surpass those of comparable endeavors, as outlined in Table 4.9. The superior performance of the proposed model can be attributed to its meticulous implementation, which includes the thoughtful selection of optimal algorithms and configurations, the utilization of advanced image processing methodologies, and the creation of an optimal training framework.

Table 4.9 Comparative Performance Analysis

| **Method** | **Performance Metrics** | **Source** |
|---|---|---|
| ConvNets | Recall = 0.9251<br>Precision = 0.8696 | (L. Zhang et al., 2016) |
| Deep Neural Networks | Recall and Precision = 0.75 | (Maeda et al., 2018) |
| Structured Prediction with the Convolutional Neural Network | Precision = 0.9178<br>Recall = 0.8812 | (Fan et al., 2018b) |
| YOLOv5s | Precision = 0.891<br>Recall = 0.512 | (Hu et al., 2021) |
| **Proposed Model (Customized YOLOv7)** | **For RDD 2022** | **Recall = 0.9545**<br>**Precision = 0.9523** |
| | **For Custom Data** | **Recall = 0.9158**<br>**Precision = 0.93** |

## 4.6. RESULTS FROM MODEL 2: CUSTOMIZED YOLOV8X-SEG MODEL

### 4.6.1. Performance Evaluation of the Proposed Model

In this research, the primary objective was to develop a highly specialized model tailored for the detection and identification of pavement cracks using instance segmentation. Utilizing the advanced capabilities of the custom-configured YOLOv8x-seg architecture, the model was adeptly designed to capture and interpret the complex realities of road surface conditions. The core of the study was a meticulously compiled dataset, featuring diverse road surfaces from Selangor and Kuala Lumpur, which provided a comprehensive representation of different crack types.

The training regimen spanned 200 epochs, a duration thoughtfully chosen to ensure a balance between effective learning and computational practicality. Additionally, a batch size of 16 was selected, optimizing the gradient estimation's accuracy while remaining within the bounds of our hardware capabilities. This batch size was ideal for allowing more frequent model updates and maintaining the representativeness of each data batch, crucial for accurately modeling the detailed characteristics of pavement cracks. The effectiveness of the model in both detection and classification tasks is illustrated in the confusion matrix presented in Figure 4.9.

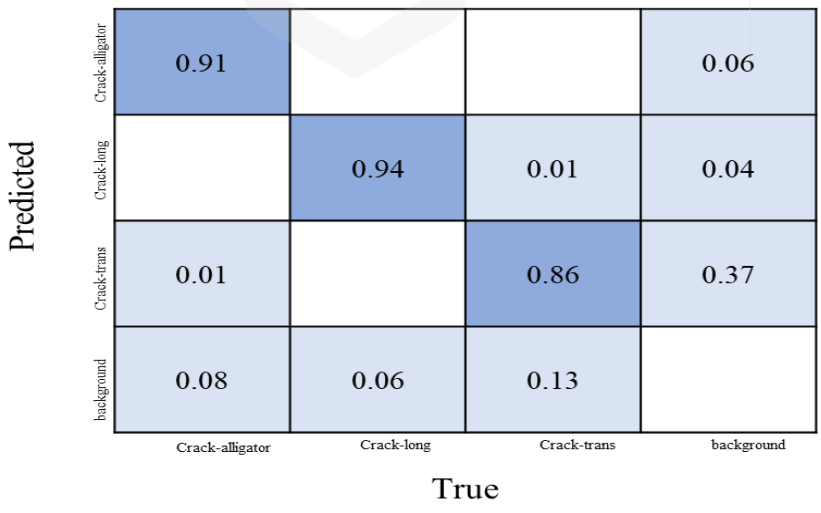



Figure 4.9 Confusion Matrix

Utilizing the data from the generated confusion matrix, critical performance indicators such as recall, accuracy, precision, and the F1 score were meticulously computed. These metrics offer an in-depth evaluation of the model's proficiency in accurately classifying different types of road cracks. They not only highlight the model's competencies but also identify potential areas for enhancement. An elaborate presentation of these metrics, including specific values for each crack category (alligator, longitudinal, and transverse), is detailed in Table 4.10.

In addition, Figures 4.10, 4.11, and 4.12 provide visual representations of the recall, precision, and F1-score, respectively, in relation to varying levels of model confidence. These visualizations effectively depict how the model's recall, precision, and F1-score fluctuate in accordance with changes in its prediction confidence, offering valuable insights into the model's performance dynamics under different scenarios.

Table 4.10 Performance Parameter Scores

| **Crack Type** | **Recall** | **Precision** | **F1-Score** | **Accuracy** |
|---|---|---|---|---|
| Alligator | 0.93 | 0.91 | 0.975 | 0.91 |
| Longitudinal | 0.95 | 0.84 | 0.97 | 0.94 |
| Transverse | 0.806 | 0.89 | 0.916 | 0.86 |
| **Average** | **0.90** | **0.88** | **0.95** | **0.90** |

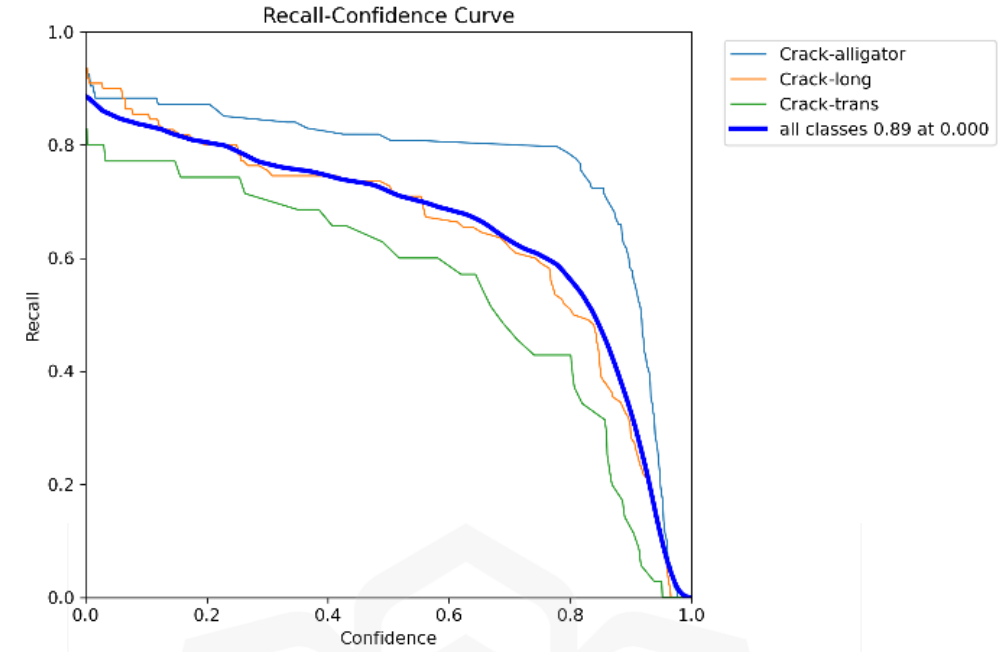



Figure 4.10 Recall-Confidence Plot

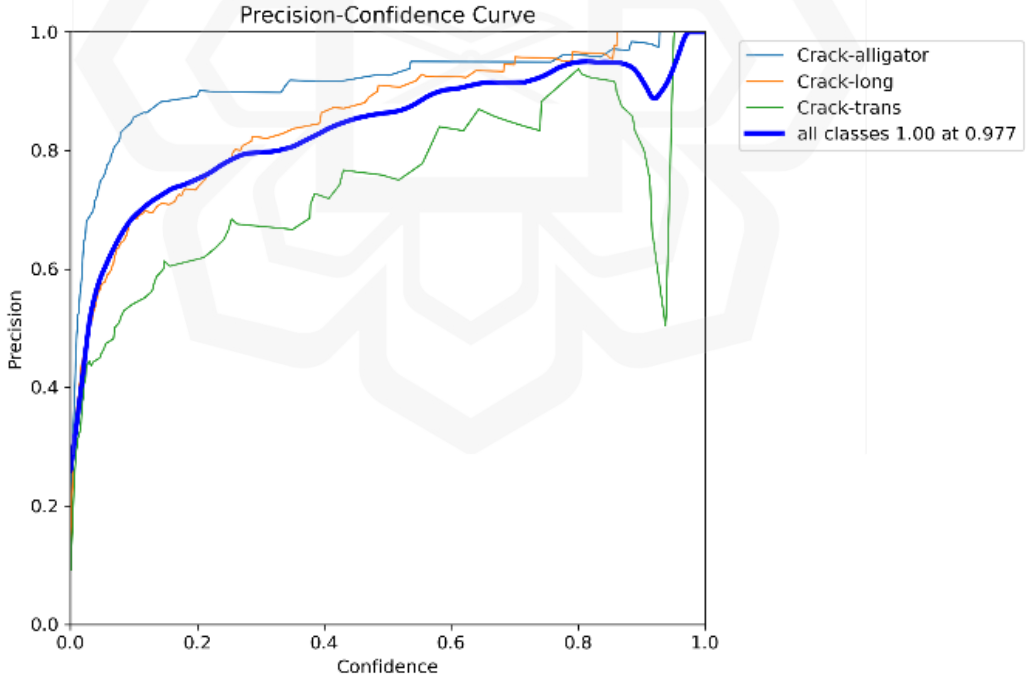



Figure 4.11 Precision-Confidence Plot

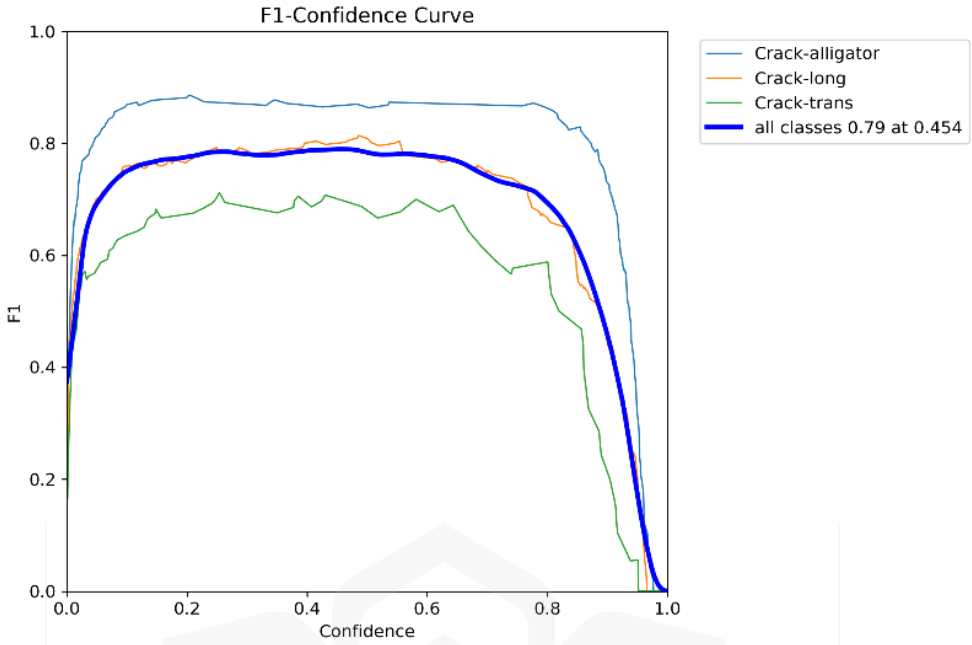



Figure 4.12 F1-score - Confidence Plot

Upon assessing the model's capabilities in object detection, the findings related to instance segmentation, particularly regarding recall, precision, and F1-score, are displayed in Figures 4.13, 4.14, and 4.15. This aspect of the study is of paramount importance, as our goal is to precisely identify and delineate every individual crack on the pavement surfaces. Utilizing the real-world road data we collected, along with the capabilities of our customized YOLOv8x-seg model, these results are instrumental in gauging the model’s effectiveness in detecting and distinctly marking each crack. This level of detailed and accurate segmentation is vital for enhancing the quality and precision of our pavement inspection processes.

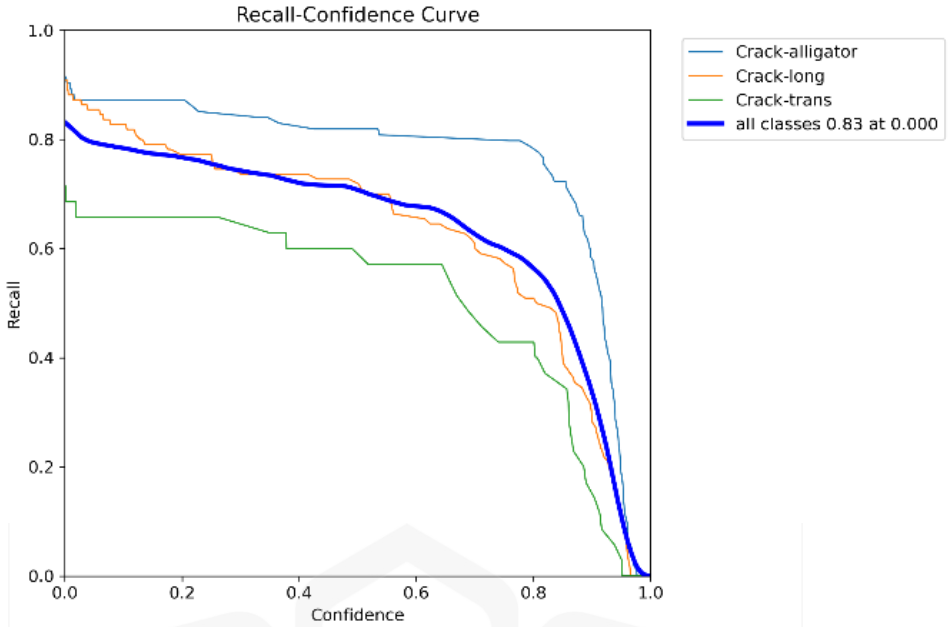



Figure 4.13 Recall-Confidence Plot (mask)

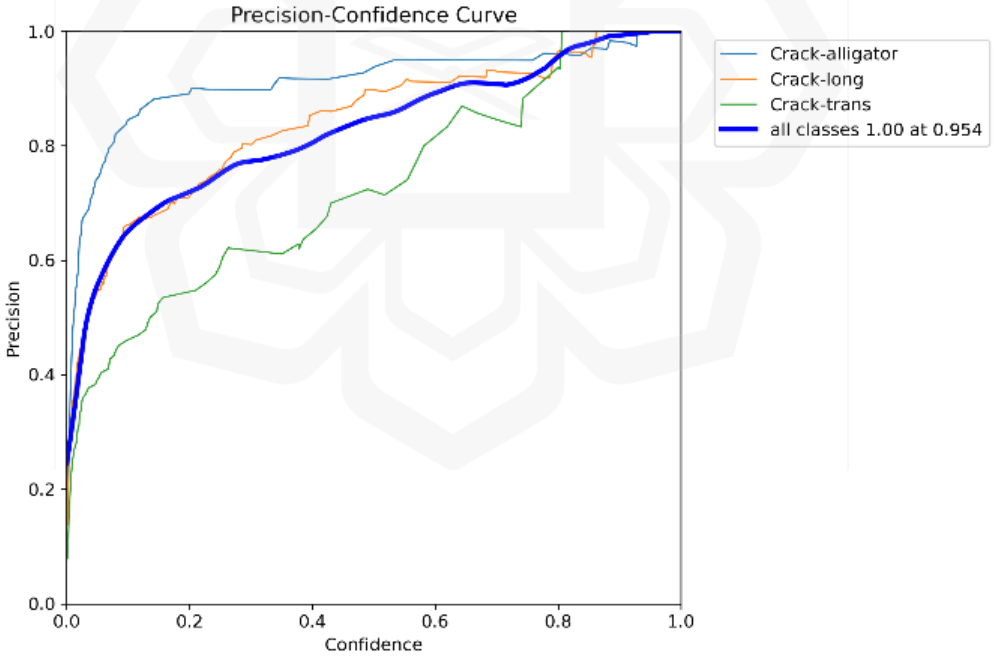



Figure 4.14 Precision-Confidence Plot (mask)

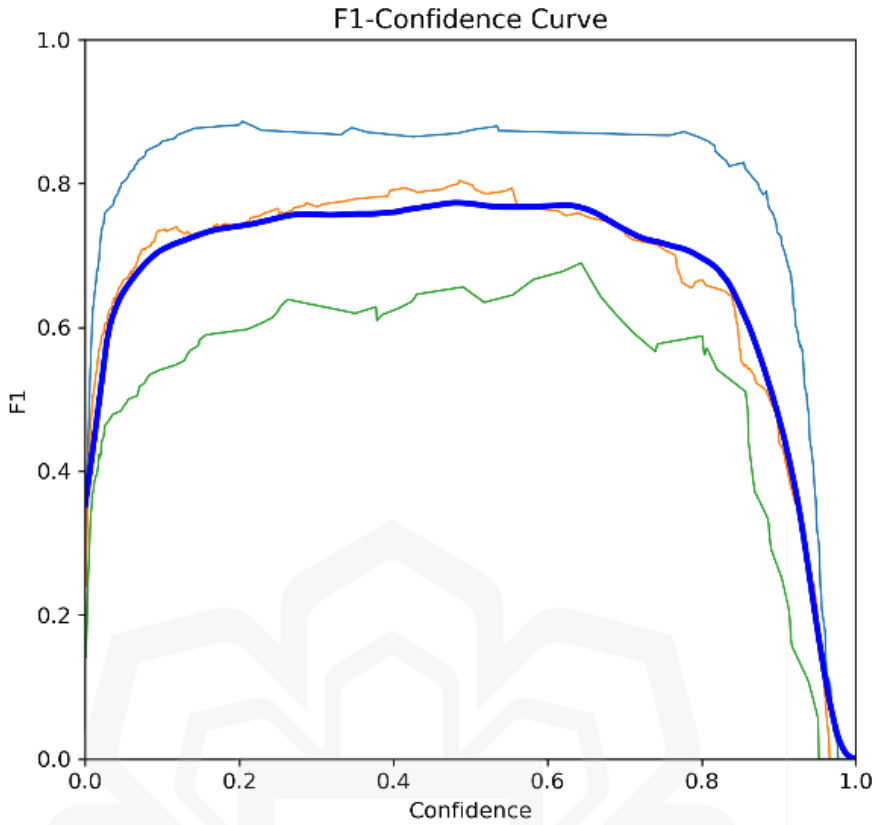



Figure 4.15 F1-score - Confidence Plot (mask)

Expanding on the evaluation of the model's performance, Figures 4.16 and 4.17 offer visual demonstrations of the model's accuracy. These figures enable a side-by-side examination of the original annotations against the model's predictions. Figure 4.16 displays a batch from the validation set, where the road cracks are precisely marked, pinpointing their exact locations and types – essentially representing the ground truth. In contrast, Figure 4.17 reveals the model's predictions applied to the same batch but without prior annotations. This comparison offers a transparent view of how effectively the model can identify and segment various types of road cracks in an unaided scenario. Such a visual juxtaposition highlights the practical applicability and reliability of the model in real-world road assessment tasks.

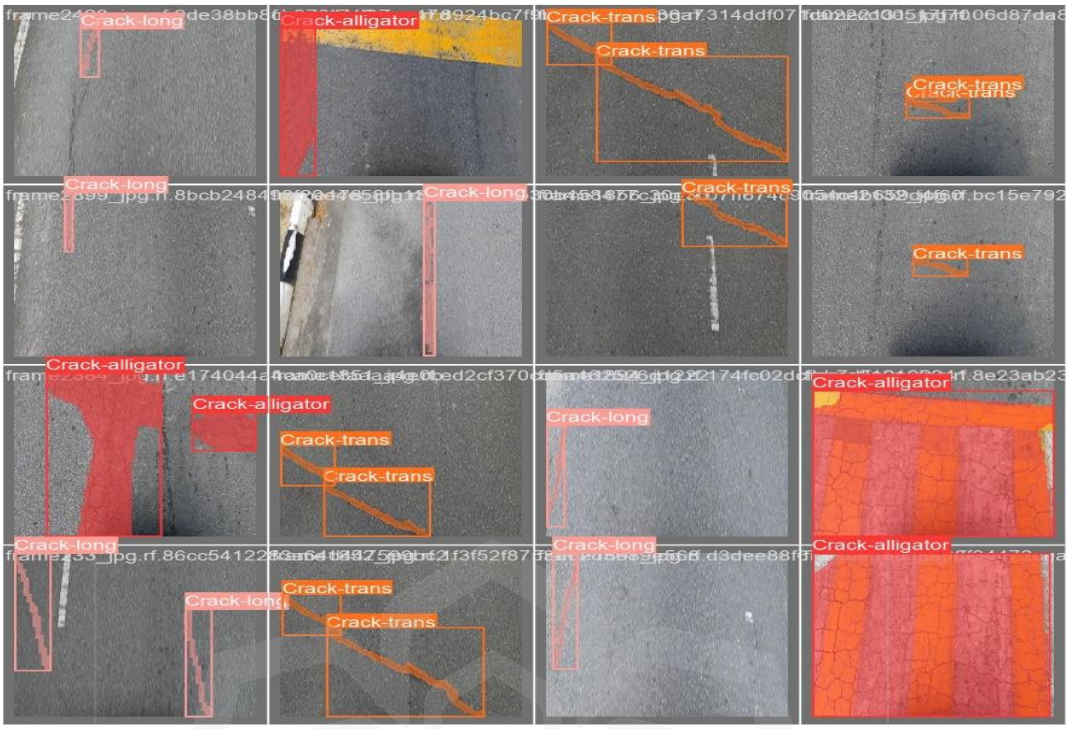



Figure 4.16 Validation batch

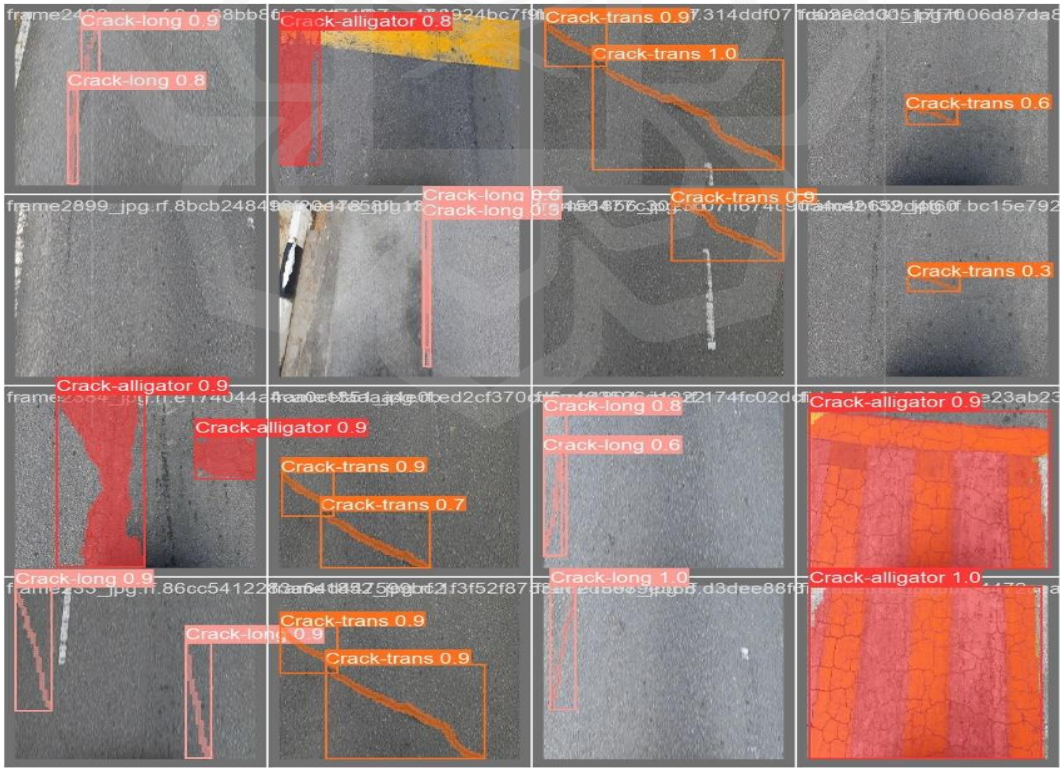



Figure 4.17 Prediction batch

### 4.6.3. Crack Sizing

As the study progresses, we delve into the detailed assessment of crack dimensions, transforming the analysis from mere visual detection to quantifiable measures. This part of the research focuses on converting the observed cracks into measurable units such as meters or square meters, providing a comprehensive view of the severity and extent of pavement degradation.

The methodology involves isolating the identified crack areas and quantifying them in terms of pixel count. This initial step is crucial as it offers a preliminary scale of the cracks. However, the real significance of this analysis lies in converting these pixel-based measurements into actual physical dimensions, thereby making the data immensely valuable for practical applications in infrastructure assessment and repair planning.

For a clear understanding of these measurements, longitudinal and transverse cracks are presented in linear meters, as depicted in Figures 4.18 and 4.19. Alligator cracks, being more area-centric, are calculated and displayed in square meters, as shown in Figure 4.20. These measurements are grounded in the previously established mathematical formulas, which consider several critical factors, including the camera's height above the road surface and the field of view capturing the road width.

This conversion from pixel measurements to real-world dimensions is not just a technical exercise but a critical step in translating data into meaningful, actionable insights for road maintenance and improvement strategies.

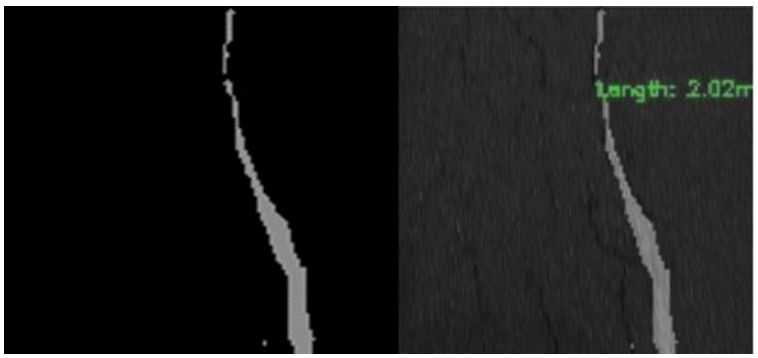


Figure 4.18 Longitudinal crack measurement from binary image

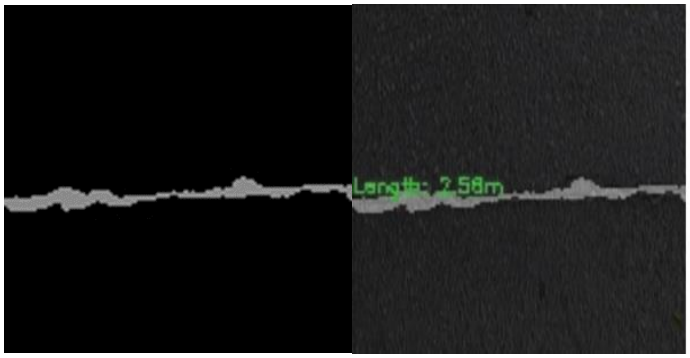


Figure 4.19 Transverse crack measurement from binary image

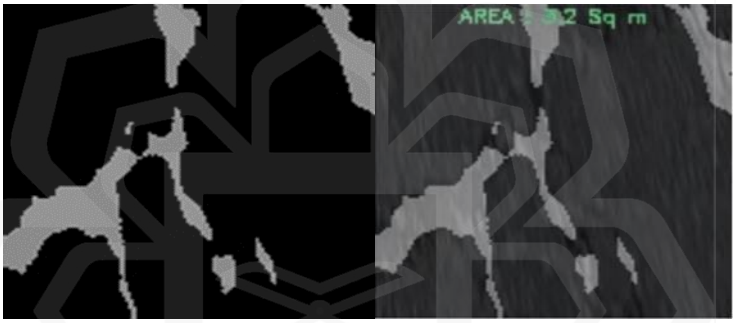


Figure 4.20 Alligator crack measurement from binary image

### 4.6.4. Benchmarking

In the field of pavement crack detection, this study introduces an advanced approach, employing a specially tailored YOLOv8x-seg model for instance segmentation. Compared to the well-regarded Deep Convolutional Neural Network Fusion Model, which similarly used self-collected real-world data, our methodology distinctly outperforms in terms of accuracy and precision. The decision to extensively train the model over 200 epochs resulted in an exceptional accuracy rate of 90%. Moreover, both the precision and recall metrics of our model exceeded those of the benchmark, as detailed in Table 4.11. This highlights the effectiveness of our model in accurately identifying and classifying real-world pavement damages.

Table 4.11 Direct benchmarking comparison

| **Features/Criteria** | **Our Research (YOLOv8x-seg)** | **Benchmarked With** (Feng et al., 2020) |
|---|---|---|
| **Methodology** | Instance Segmentation using Customized YOLOv8x-seg | Deep Convolutional Neural Network Fusion Model |
| **Data Source** | Custom Collected Data | Custom Collected Data |
| **Epochs** | 200 | 100 |
| **Accuracy (%)** | 90 | 86 |
| **Precision (%)** | 88 | 81 |
| **Recall (%)** | 90 | 82 |

## 4.7. RESULTS FROM MODEL 3: ADVANCED HYBRID DEEP LEARNING MODEL

### 4.7.1. Segmentation Model Performance Analysis

The performance metrics of our segmentation model are visually depicted through a series of figures, each demonstrating the evolution of the model's accuracy, precision, recall, Jaccard coefficient, and Dice coefficient over the course of training and validation phases.

1. **Model Accuracy Across Epochs:** The accuracy graph demonstrates that the segmentation model maintains a high level of accuracy throughout the training process shown in Figure 4.21. During training (shown in blue), the accuracy begins at around 80% and quickly ascends, stabilizing just below 95%. The validation accuracy (shown in orange) closely mirrors the training accuracy, indicating consistent performance and generalization of the model to unseen data.

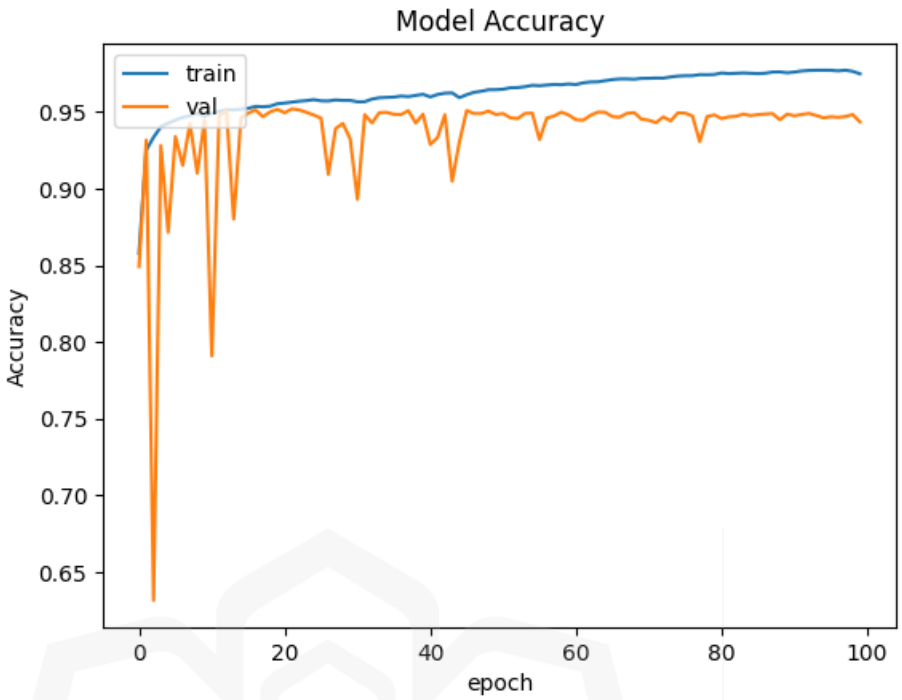



Figure 4.21 Model Accuracy Plot

2. **Model Precision Across Epochs:** Precision, representing the proportion of true positive predictions, remains consistently high across epochs for both training and validation, with both lines converging at approximately 95% shown in Figure 4.22. This metric underscores the model's capacity to correctly identify crack segments with minimal false positives.

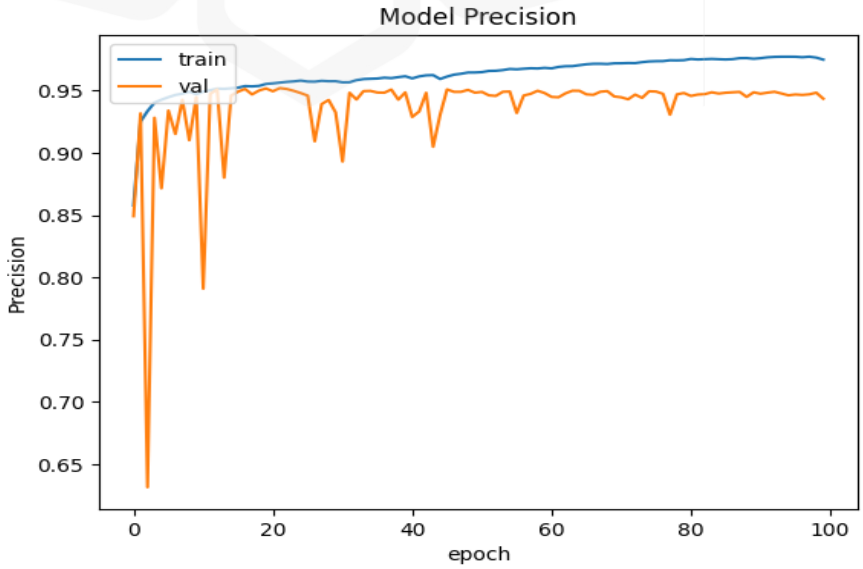



Figure 4.22 Model Precision Plot

3. **Model Recall Across Epochs:** Recall, which measures the model's ability to detect all relevant instances, is depicted to stabilize at near 90% for both training and validation, implying the model's robust detection capabilities shown in Figure 4.23.

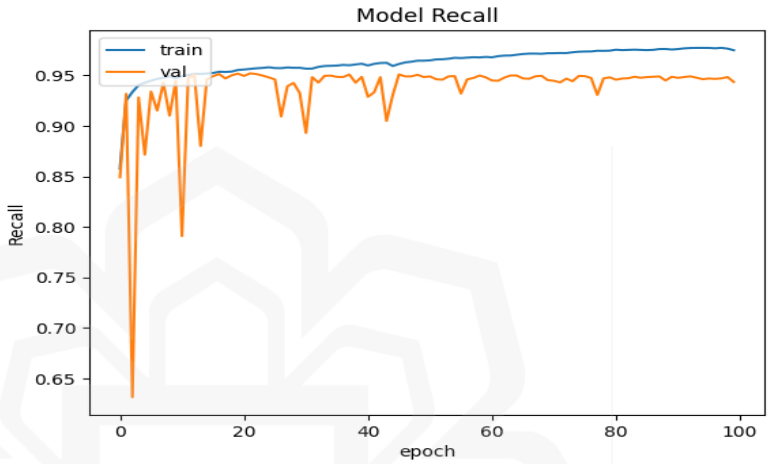



Figure 4.23 Model Recall Plot

4. **Model Jaccard Coefficient Across Epochs:** The Jaccard Coefficient, or Intersection over Union, graph indicates an upward trend, plateauing around 80% for both training and validation shown in Figure 4.24. This reflects the model's consistent and reliable segmentation.

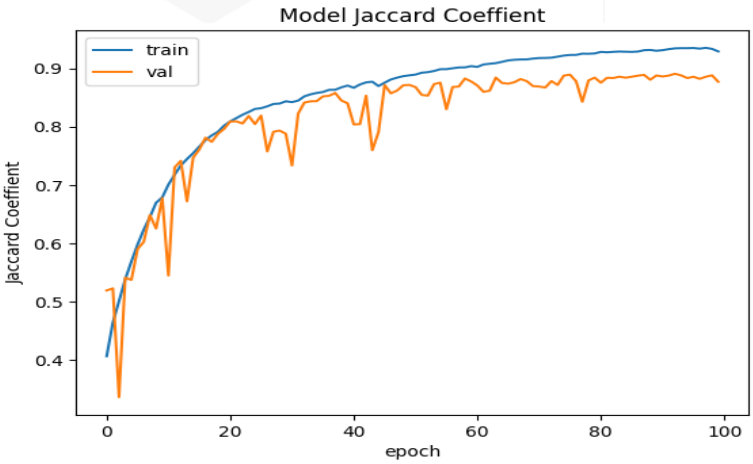



Figure 4.24 Model Jaccard Coefficient Plot

5. **Model Dice Coefficient Across Epochs:** Similar to the Jaccard Coefficient, the Dice Coefficient also exhibits an upward trend, stabilizing slightly above the Jaccard Coefficient, indicating a high degree of overlap and similarity between the predicted segments and ground truth shown in Figure 4.25.

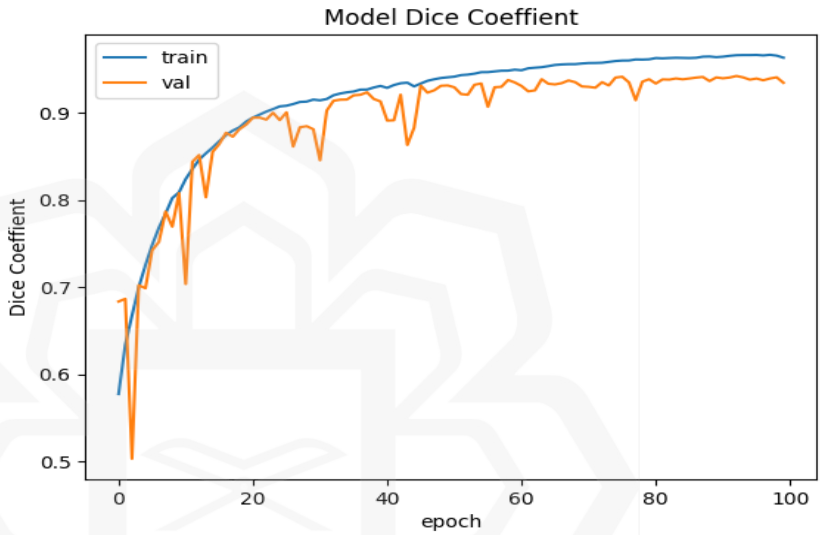



Figure 4.25 Model Dice Coefficient Plot

In addition to these graphical representations, the segmentation model's performance is quantitatively summarized in Table 4.12, detailing the key metrics for both the background and crack classes.

Table 4.12. Quantitative Metrics for Segmentation Model

| Metric | Value |
|---|---|
| Mean IoU | 0.8388889 |
| Mean Dice Coefficient | 0.8256968848551859 |
| Hausdorff Distance for Background | 3.2126949574078867 |
| Hausdorff Distance for Crack | 3.2126949526675244 |

The Mean IoU of 0.8388889 reflects the model's average efficacy in overlapping the predicted and actual areas, while the Mean Dice Coefficient of 0.8256968848551859 further corroborates the model's consistent performance in segmentation. The Hausdorff Distance for both Background and Crack is approximately 3.21, illustrating the precision with which the model delineates the segmented areas from the background. These values indicate a high level of precision in the model's ability to segment and identify cracks, validating the model's application in real-world pavement analysis scenarios.

Figures 4.26, 4.27, and 4.28 depict the segmentation results for alligator, transverse, and longitudinal cracks, respectively. These visual outcomes not only validate the model's precision but also underscore its applicability in practical pavement evaluation scenarios. Through meticulous image analysis, the model demonstrates a high degree of accuracy in identifying the intricate details of crack patterns, which is essential for the effective assessment and subsequent treatment of pavement distresses.

- ***Alligator Crack Segmentation:*** Figure 4.26 showcases the original grayscale image of an alligator crack (left) and the corresponding segmented image (right). The original image presents a complex network of interconnected cracks characteristic of alligator cracking, named for its resemblance to alligator skin. The segmented image reveals the model's precision in isolating and highlighting these intricate crack patterns, indicating the model's robust capability to detect and delineate extensive crack formations within the pavement surface.

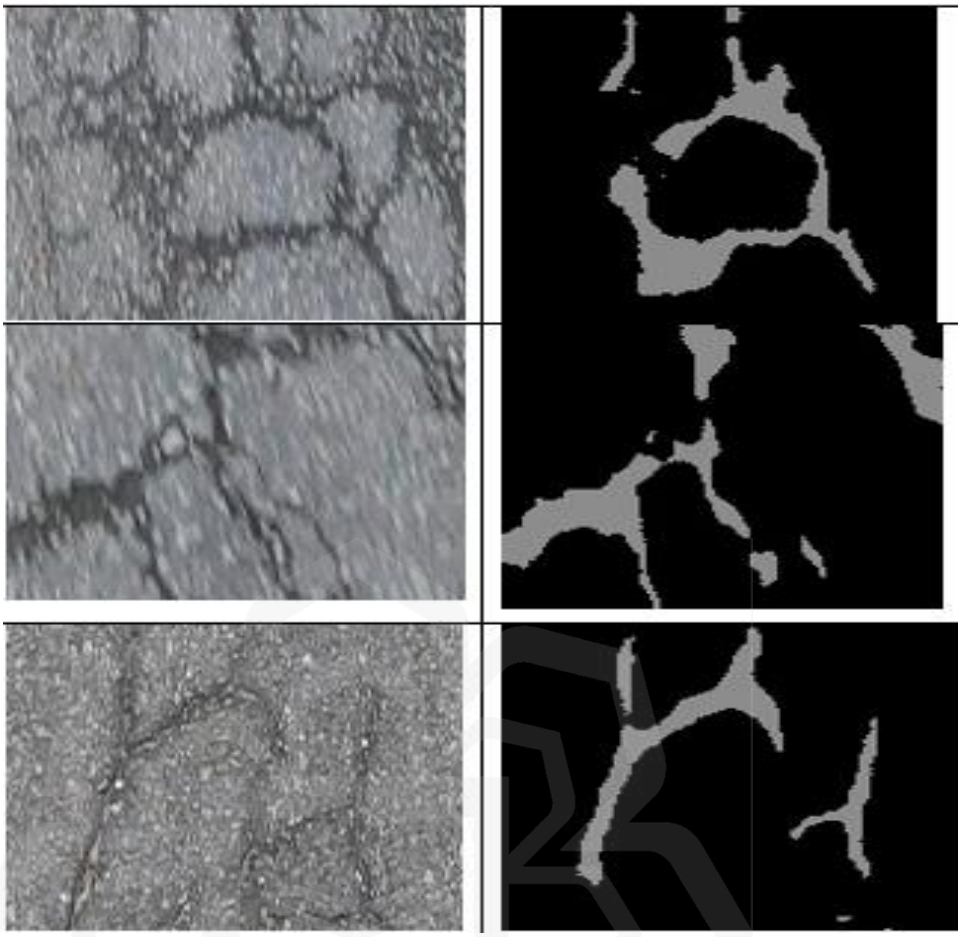


Figure 4.26 Original and Segmented image of alligator crack

- ***Transverse Crack Segmentation:*** Figure 4.27 illustrates a transverse crack, which typically runs perpendicular to the direction of traffic. The original image (left) depicts a singular, prominent line of damage across the pavement, while the segmented image (right) captures the clear and uninterrupted linear crack form. This demonstrates the model's effectiveness in identifying and segmenting transverse cracks, ensuring that such linear discrepancies on the road surface are accurately captured for further analysis.

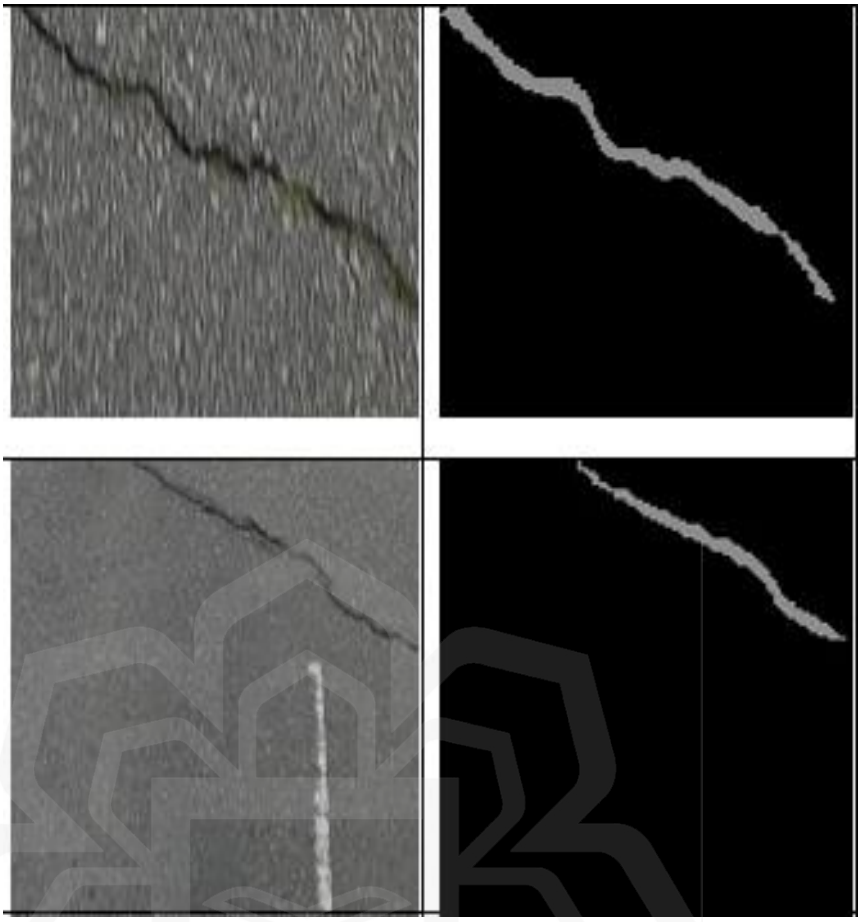


Figure 4.27. Original and Segmented image of transverse crack

- ***Longitudinal Crack Segmentation:*** In Figure 4.28, the longitudinal crack, running parallel to the direction of traffic, is displayed. The raw image (left) displays a thin, elongated crack, which may be indicative of initial pavement distress. The segmented counterpart (right) accurately traces the length of the crack, providing a distinct and isolated representation of the pavement flaw. This segmentation underlines the model's capability to recognize and segregate longitudinal cracks, which is critical for structural assessments and maintenance planning.

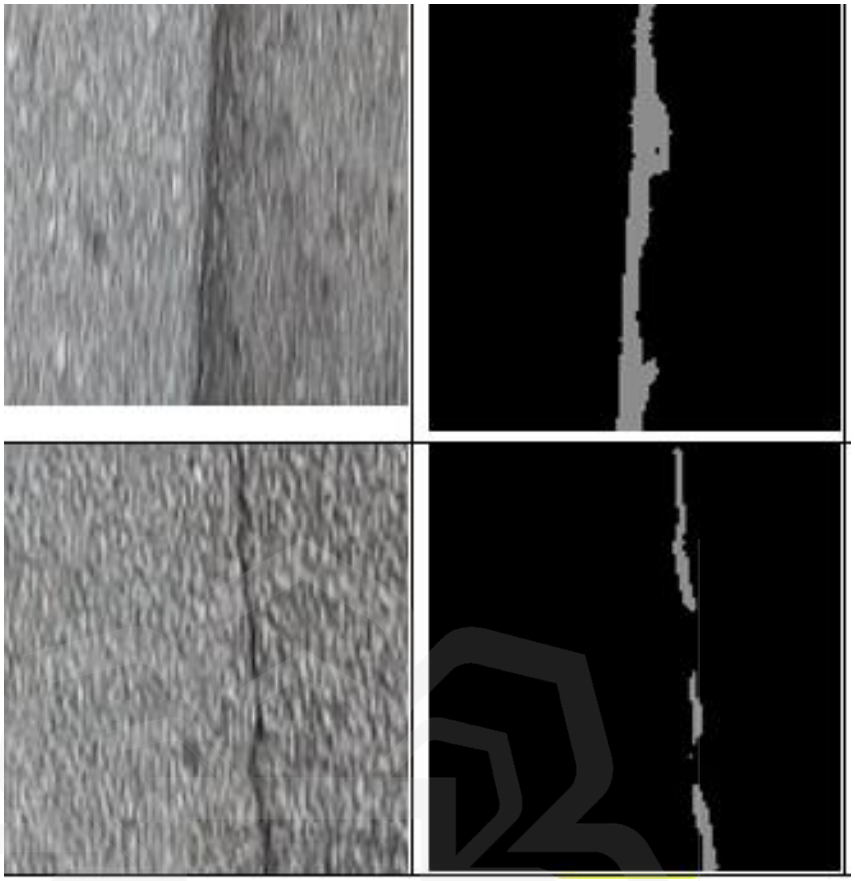


Figure 4.28. Original and Segmented Image of Longitudinal Crack

### 4.7.2. Classification Model Performance Analysis

Following the segmentation of pavement cracks, the next pivotal step in our study is the classification of these segmented images. The classification model's ability to distinguish between different types of cracks is essential for proper pavement assessment and subsequent maintenance decision-making. This section delves into the performance analysis of the classification model, which is based on a Residual block with a Modified Attention mechanism, ensuring that each crack type is accurately recognized and categorized.

Our classification report in Table 4.13 provides a detailed breakdown of the model's precision, recall, and F1-score for each crack type, alongside the support, which denotes the number of instances for each class within the test dataset. This report is instrumental in understanding the model's performance across different classes of pavement cracks:

- For alligator cracks (class 0), the model exhibits a precision of 0.84 and a recall of 0.96, resulting in an F1-score of 0.90. This high recall indicates the model's strong ability to detect almost all alligator cracks present in the dataset.
- In identifying longitudinal cracks (class 1), the model maintains a precision of 0.89 and a recall of 0.88, with an F1-score of 0.885, showcasing its reliable classification despite the complex nature of these cracks.
- The transverse cracks (class 2) are classified with a precision of 0.87, but a recall of 0.80, leading to an F1-score of 0.83, reflecting a slightly more challenging detection scenario for this type of crack.

The overall accuracy of the classification model stands at 0.85, which signifies that the model correctly identifies the crack type in 85% of the cases across the dataset. The macro and weighted averages for precision, recall, and F1-score are consistent, hovering around the 0.85 mark, which further indicates the model's balanced performance across all crack types.

Table 4.13 Classification Report for Residual block with Modified Attention mechanism

| | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| 0 | 0.84 | 0.96 | 0.90 | 114 |
| 1 | 0.89 | 0.88 | 0.885 | 135 |
| 2 | 0.87 | 0.80 | 0.83 | 121 |
| accuracy | 0.85 | | | 370 |
| macro avg | 0.85 | 0.85 | 0.85 | 370 |
| weighted avg | 0.85 | 0.85 | 0.84 | 370 |

The confusion matrix in Figure 4.29 provides a visual and quantitative representation of the classification model's performance. The matrix illustrates the number of true positives (diagonal entries) where the model's predictions align with

the actual crack types. It also sheds light on the instances of misclassification (off-diagonal entries):

1. The majority of alligator cracks are correctly classified (110 out of 114), with a few instances (4) being confused with longitudinal cracks.
2. Longitudinal cracks have a higher misclassification rate, with some being mistaken for alligator (9) or transverse cracks (14).
3. Transverse cracks also see some confusion, primarily with longitudinal cracks (18), indicating an area where the model might benefit from further training or refinement.

The classification after segmentation indicates that the segmentation quality has a direct influence on the classification outcomes. High-quality segmentation is likely to yield more accurate classification results, as evidenced by the high recall scores for alligator cracks. The model's performance in distinguishing longitudinal and transverse cracks, while commendable, suggests potential for improvement in differentiating between these types with similar visual features.

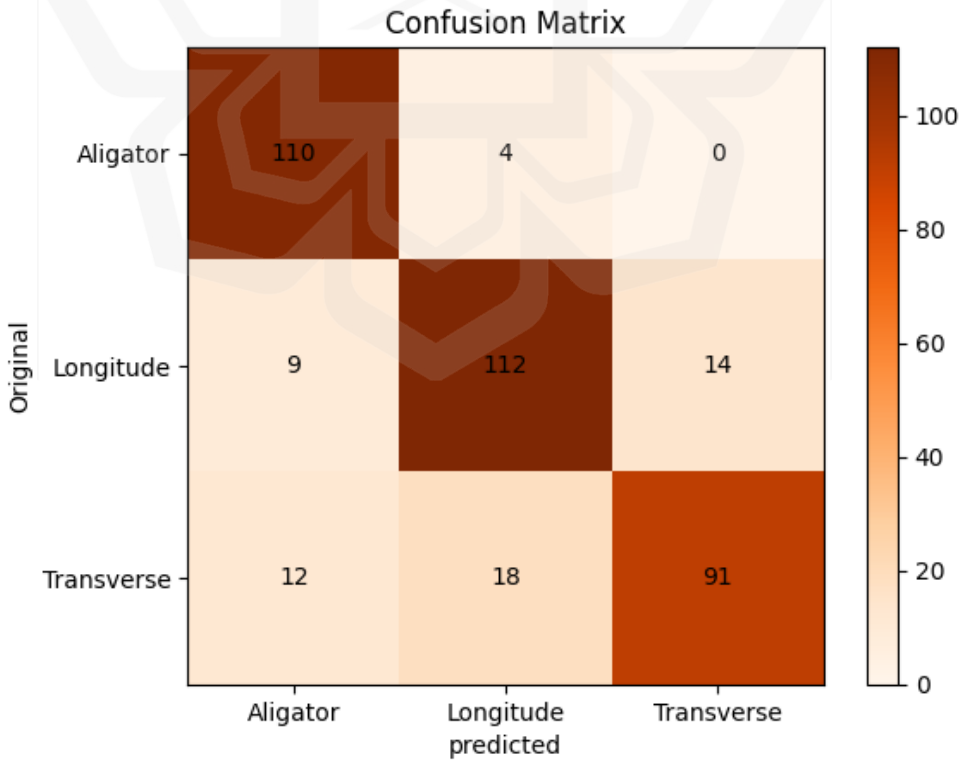



Figure 4.29 Confusion Matrix for Residual block with Modified Attention Mechanism

### 4.7.3. Image Concatenation Results

The results of the crack analysis using the image concatenation method are presented below. This section details the original images, the concatenated and segmented images, and the predicted results for each type of crack: Alligator, Longitude, and Transverse. For each crack type, we began with two original images that were concatenated horizontally or vertically, depending on the orientation of the cracks. The concatenated images were then segmented using our deep learning model to highlight the crack areas. The visualization of the original, concatenated, and segmented images provided a clear comparison, allowing for an assessment of the model's performance as shown in Figure 4.30, Figure 4.31 and Figure 4.32.

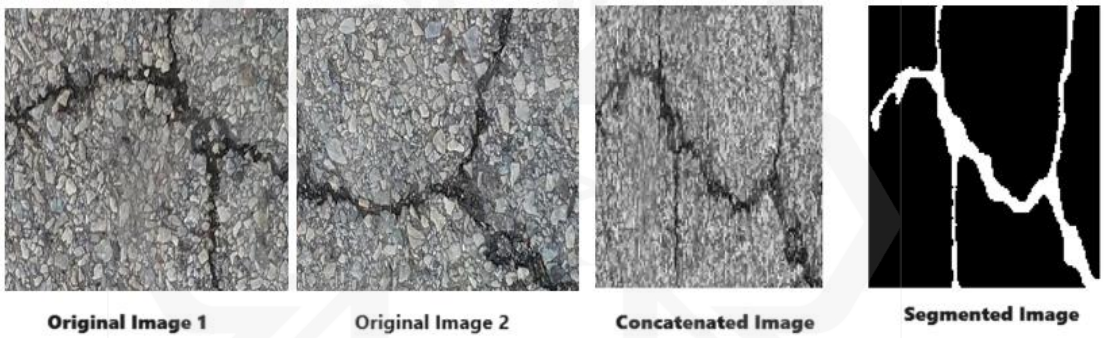



Figure 4.30 Concatenation of Alligator Crack Image Frames

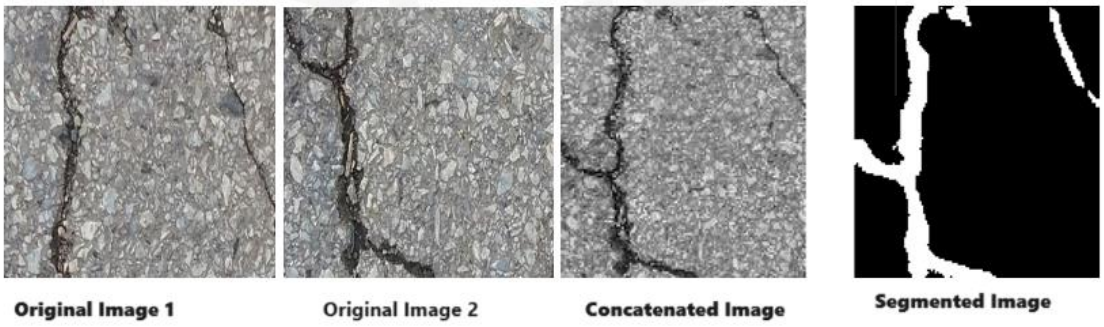



Figure 4.31 Concatenation of Longitudinal Crack Image Frames

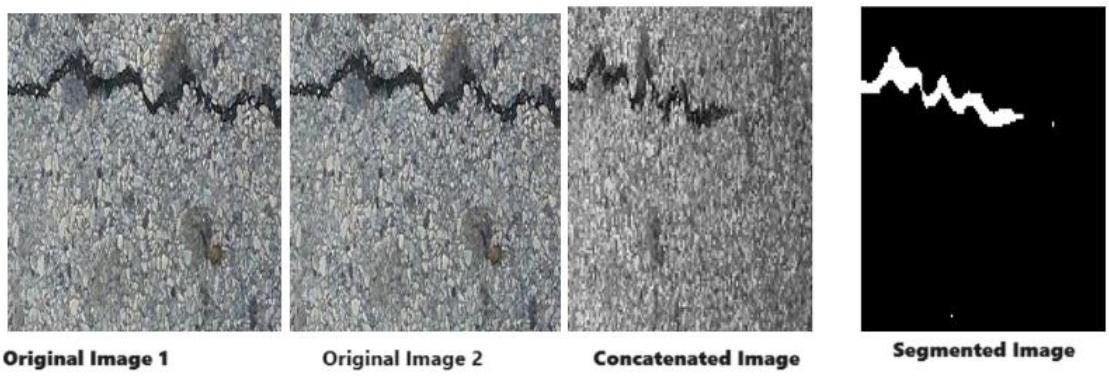



Figure 4.32 Concatenation of Transverse Crack Image Frames

### 4.7.4. Crack Measurement and Classification Results

The quantitative analysis of pavement cracks is a critical component in the assessment of road infrastructure. This segment of the study transitions from the preliminary visual detection to the meticulous sizing of cracks, where the dimensions are not only identified but also measured with precision. This shift is crucial for understanding the severity and potential impact of the pavement damage observed.

The process begins with the isolation of the crack areas identified in the segmentation phase. These regions are then quantified using pixel counts, providing an initial scale of the damage. The subsequent conversion of these pixel measurements into real-world units—meters and square meters—elevates the analysis from a simple count to a meaningful metric that is vital for infrastructure evaluation and maintenance planning.

In this context, longitudinal and transverse cracks are measured in linear meters. This linear measurement is essential for planning repair processes that often involve filling or sealing cracks along their length. Alligator cracks, known for their extensive and area-covering nature, are quantified in square meters. The area measurement of alligator cracks is crucial for assessing the extent of surface deterioration and determining the need for more extensive repair or replacement interventions.

Figures 4.33, 4.34, and 4.35 illustrate the result of this conversion process, where each type of crack is measured and displayed in its respective units, offering a clear and tangible representation of the crack dimensions.

- ***Longitudinal Crack:*** Figure 4.33 presents a detailed view of a longitudinal crack. The crack runs parallel to the road's direction, and its measurement in meters reflects the extent of the crack that could potentially impact the structural integrity of the pavement.

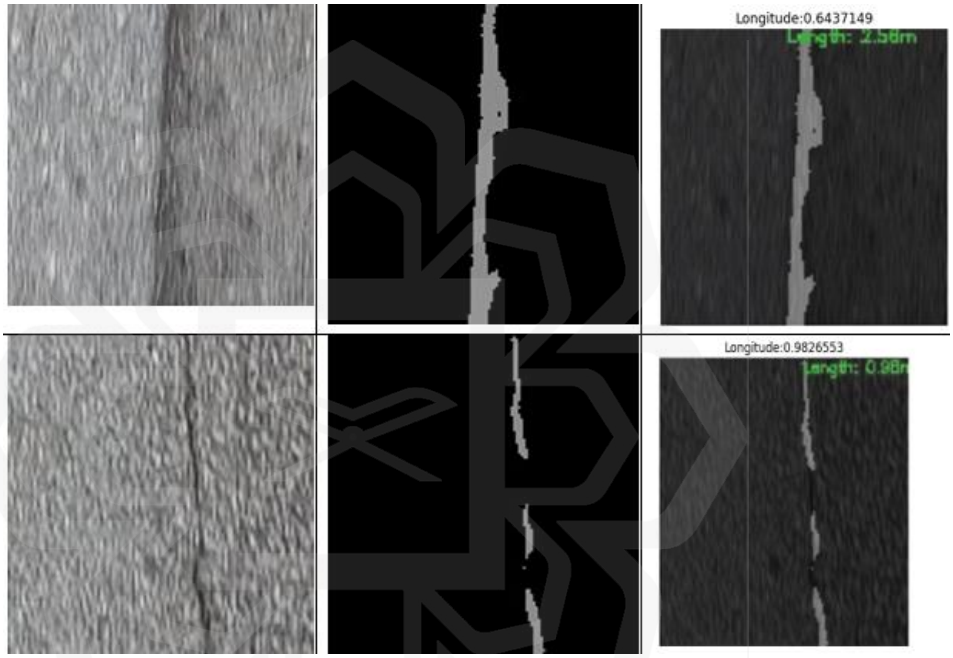



Figure 4.33 Longitudinal Crack Classification and Measurement

- ***Transverse Crack:*** In Figure 4.34, a transverse crack is showcased. Perpendicular to the roadway, the measured length of this type of crack provides insight into the potential for water infiltration and the urgency of repair to prevent further pavement deterioration.

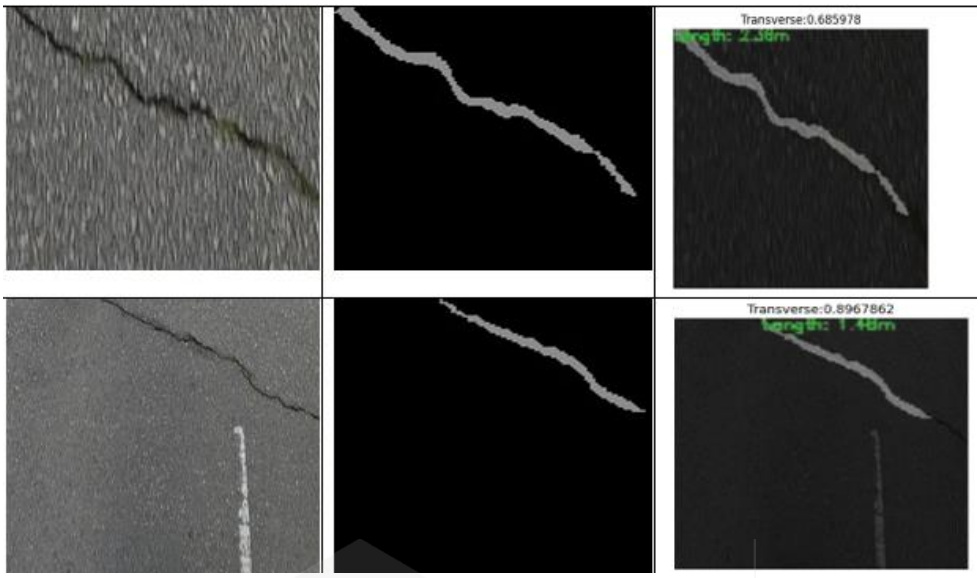



Figure 4.34 Transverse Crack Classification and Measurement

- ***Alligator Crack:*** Figure 4.35 depicts the complex pattern of alligator cracking. The area measurement in square meters illustrates the significant surface area affected, which can be indicative of foundational issues requiring substantial rehabilitation efforts.

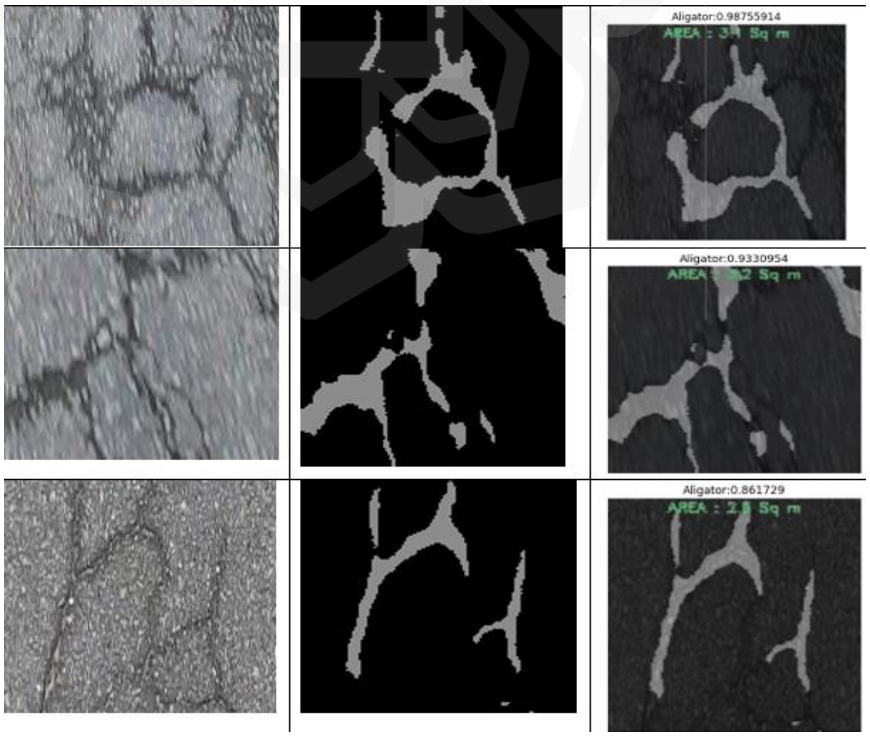



Figure 4.35 Alligator Crack Classification and Measurement

Integrating the precise measurements provided in these figures with the detailed segmentation and classification data from the previous sections, this study provides a comprehensive and quantifiable analysis of pavement cracks. Such detailed assessment is indispensable for developing targeted and cost-effective road maintenance and improvement strategies, ensuring road safety and longevity.

### 4.7.5. Validation of Crack Measurement Results

The results of the comparative analysis of distance measurements at different camera heights are presented below. Table 4.14 highlights the actual measured lengths between the points, the detected lengths at each camera height, and the corresponding variations from the actual measurements. This analysis helps in understanding the impact of camera height on the accuracy of the detected lengths.

Table 4.14 Comparative Analysis of Distance Measurements at Different Camera Heights

| Points | Actual Measured Length (m) | Detected Length (m) at 1.57m | Variation at 1.57m | Detected Length (m) at 1.60m | Variation at 1.60m | Detected Length (m) at 1.63m | Variation at 1.63m |
|---|---|---|---|---|---|---|---|
| A-B | 0.25 | 0.27 | +0.020 | 0.255 | +0.005 | 0.238 | -0.012 |
| A-C | 0.50 | 0.543 | +0.043 | 0.508 | +0.008 | 0.485 | -0.015 |
| A-D | 1.00 | 1.08 | +0.080 | 1.027 | +0.027 | 0.976 | -0.024 |
| A-E | 1.50 | 1.605 | +0.105 | 1.513 | +0.013 | 1.474 | -0.026 |

The study demonstrates that camera height significantly influences the accuracy of length detection in computer vision systems. Among the different heights tested, a camera height of 1.60m yielded the most accurate results, showing minimal

variation from the actual measurements as shown in Figure 4.36. This height was therefore used in our research for data collection due to its optimal performance in accurately capturing the crack lengths.

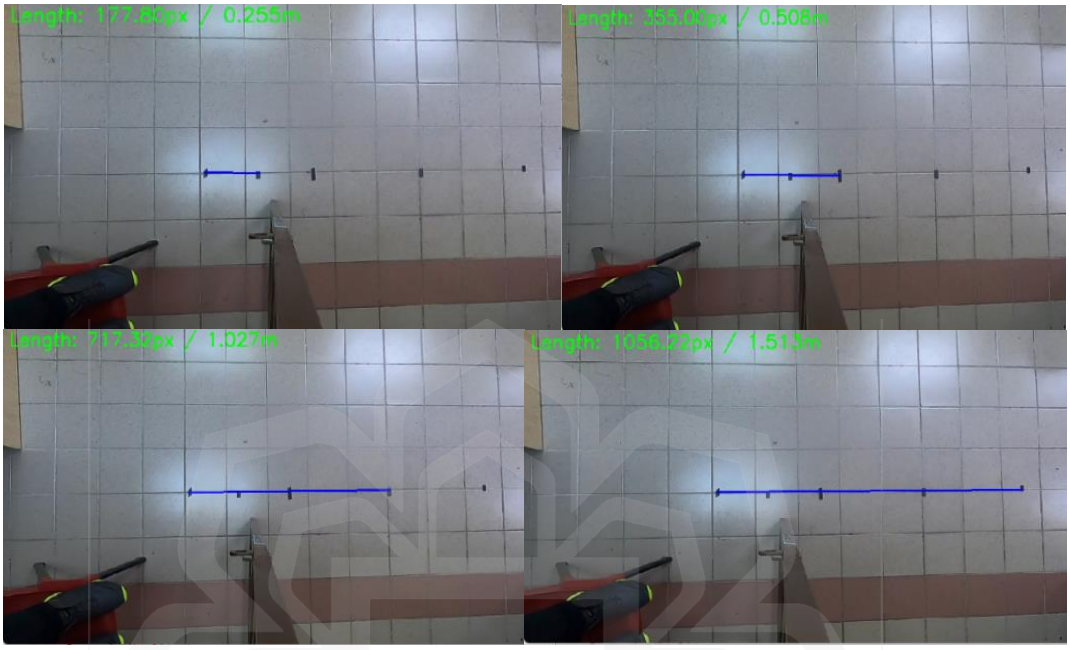


Figure 4.36 Detected Length (Camera Height 1.60m)

When the camera height was decreased to 1.57m shown in Figure 4.37, the detected lengths were generally overestimated, as indicated by positive variations. Conversely, increasing the camera height to 1.63m resulted in the detected lengths being underestimated, as shown by negative variations shown in Figure 4.38. This trend highlights the importance of selecting an appropriate camera height, as both lower and higher heights can lead to significant measurement inaccuracies. Thus, the validation process confirms that our choice of a 1.60m camera height was indeed optimal for our developed data acquisition setup, ensuring accurate and reliable measurements.

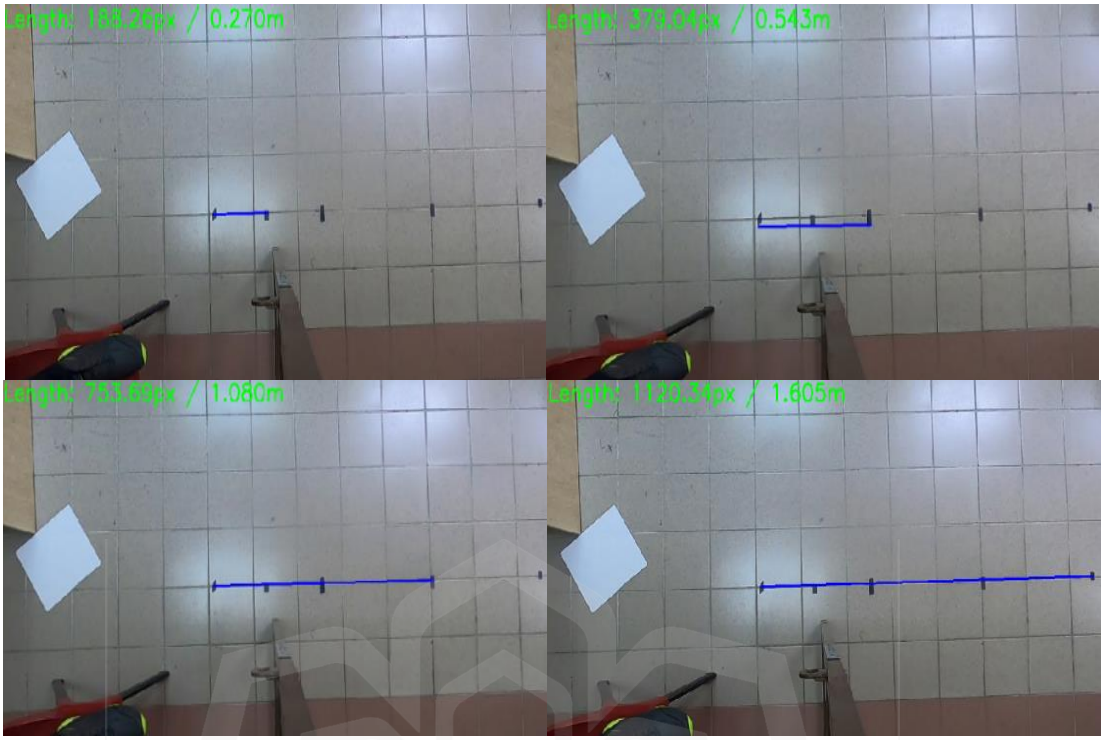



Figure 4.37 Detected Length (Camera Height 1.57m)

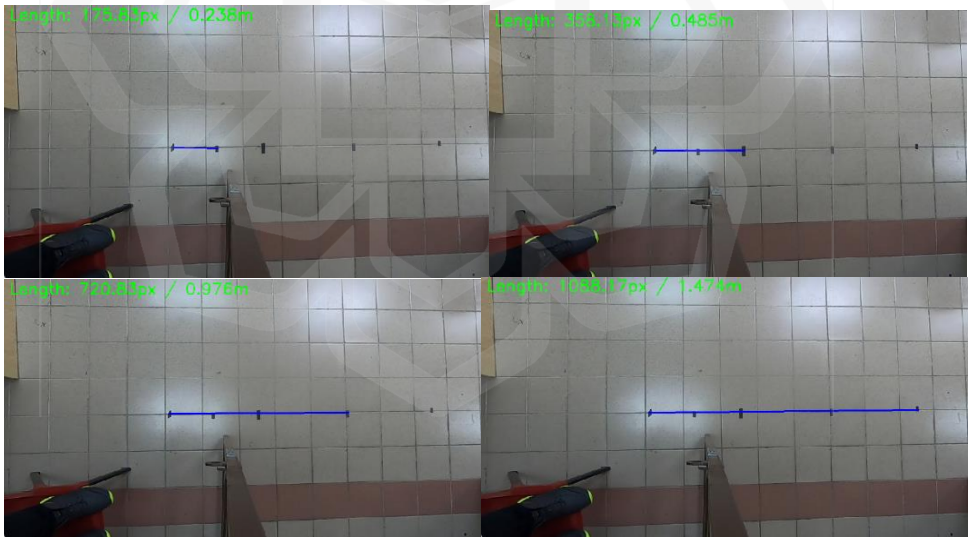



Figure 4.38 Detected Length (Camera Height 1.63m)

### 4.7.6. Benchmarking

In the domain of pavement crack detection, a benchmarking analysis between our Advanced Hybrid Deep Learning Model and the CrackNet Network Model (as

referenced in the cited publication) offers insightful comparisons. Both studies utilized custom-collected data, providing a level playing field for comparing methodologies and outcomes. Our research harnessed the combined power of Deep Gradient ResNet and a Residual block with a Modified Attention mechanism. This innovative approach allowed for intricate pattern recognition and focused attention on relevant features, crucial for accurately detecting and classifying pavement cracks as shown in Table 4.15.

Table 4.15 Benchmarking

| **Features/Criteria** | | **Our Research (Advanced Hybrid Deep Learning Model)** | **Benchmarked With** (Y. Zhao et al., 2023b) |
|---|---|---|---|
| **Methodology** | | Deep Gradient ResNet and Residual block with Modified Attention mechanism | CrackNet Network Model |
| **Data Source** | | Custom Collected Data | Custom Collected Data |
| **Epochs** | | 200 | 100 |
| **Precision** | Alligator Crack | 0.84 | 0.778 |
| | Longitudinal Crack | 0.89 | 0.867 |
| | Transverse Crack | 0.87 | 0.839 |
| **Recall** | Alligator Crack | 0.96 | 0.772 |
| | Longitudinal Crack | 0.88 | 0.849 |
| | Transverse Crack | 0.80 | 0.868 |

The extended training period of 200 epochs for our model, compared to 100 epochs for the CrackNet, likely contributed significantly to the enhanced precision and recall rates observed in our results. This extended training regimen likely allowed for a deeper learning and more nuanced understanding of the crack types and their characteristics.

When examining precision, which assesses the model's accuracy in identifying positive instances, our research consistently outperformed the CrackNet model across all crack types. For alligator cracks, our model achieved a precision of 0.84, surpassing the CrackNet's 0.778. This indicates a higher accuracy in correctly classifying alligator cracks. Similarly, for longitudinal cracks, our model reached a precision of 0.89 against 0.867 by CrackNet, and for transverse cracks, it achieved 0.87 compared to CrackNet's 0.839, further demonstrating our model's superior ability in correctly identifying these crack types.

In terms of recall, which measures the model's capability to capture all relevant instances, our model showed a notable edge in detecting alligator cracks with a recall rate of 0.96, significantly higher than the 0.772 achieved by CrackNet. This implies that our model was more effective in identifying all instances of alligator cracks. For longitudinal cracks, our model attained a recall of 0.88, again outperforming the CrackNet's 0.849. However, in the case of transverse cracks, our model's recall was slightly lower at 0.80, compared to 0.868 by CrackNet, suggesting some room for improvement in detecting this particular type of crack.

Overall, the comparative analysis underscores the efficacy of our Advanced Hybrid Deep Learning Model, particularly in its precision in identifying various types of cracks and its ability to detect the majority of these instances. The results highlight the benefits of an extended training period and a sophisticated combination of deep learning techniques, reinforcing the model's potential as a robust tool for precise and comprehensive pavement inspection and maintenance planning.

## 4.8. DISCUSSION

The results of this research, particularly through the evaluation of the customized YOLOv7, YOLOv8x-seg, and Advanced Hybrid Deep Learning models, demonstrate significant advancements in the field of pavement crack detection. In comparison to traditional methods and existing deep learning models, the proposed models offer enhanced accuracy, precision, and recall. These improvements highlight the

effectiveness of the methodological innovations implemented in this study, such as the use of pixel-level segmentation and advanced instance segmentation techniques.

One of the key contributions of this research is the ability of the YOLOv8x-seg model to achieve high accuracy in detecting and classifying multiple types of cracks, including alligator, longitudinal, and transverse cracks. This model's results, particularly in terms of precision and recall, consistently outperformed other benchmark models, such as the CrackNet and CNN Fusion models. The precision of 0.88 and recall of 0.90 for the alligator crack type, as compared to the 0.778 and 0.772 achieved by the CrackNet model, illustrates the superiority of our model in handling real-world pavement crack detection tasks. This suggests that the use of extensive training (200 epochs) and a robust dataset, including both custom-collected and publicly available data, significantly improved the model's generalization capability.

Further analysis of the Advanced Hybrid Deep Learning model highlights its strengths in combining the Deep Gradient ResNet and Modified Attention mechanism, resulting in exceptional performance, particularly for more complex cracks such as alligator and longitudinal types. The model's ability to maintain a high recall rate (0.96 for alligator cracks) showcases its sensitivity in capturing critical instances of pavement distress, which is a vital aspect of road infrastructure assessment. This sensitivity can be attributed to the multi-scale feature extraction and attention-guided learning implemented in the model architecture. Comparatively, the model performed on par or better than the existing models, demonstrating its capacity to handle diverse road conditions and various crack patterns.

In addition to model performance, this research also makes a substantial contribution by providing a methodology for converting pixel-based crack detection into real-world measurements. By integrating advanced image processing techniques with deep learning models, this study offers a scalable approach for measuring crack dimensions in meters and square meters, providing essential information for road maintenance planning. This capability directly addresses the limitations of many existing studies, which often stop at crack detection without offering quantifiable metrics for road repair strategies.

The benchmarking analysis further underscores the contributions of this study. The proposed models not only surpass traditional approaches but also push the boundaries of current deep learning models. The models developed in this research excel in both detection and characterization of cracks, providing a comprehensive solution that can be adapted for use in real-world applications. This discussion highlights the practical impact of the models and methodological innovations, reinforcing their potential to improve road safety, reduce maintenance costs, and contribute to the advancement of the field of pavement crack analysis.

## 4.9. SUMMARY

This chapter presented a concise evaluation of the results from the models developed in this research, specifically addressing Objective 4: evaluating model performance. The customized YOLOv7, YOLOv8x, and the advanced hybrid models were tested rigorously and benchmarked against established datasets. Each model demonstrated high accuracy and made significant contributions to the automated analysis of pavement cracks.

The calibration and data collection setup, featuring the GoPro Hero 8 camera, were vital in capturing high-quality images, which ensured reliable model training. Additionally, the integration of external datasets like RDD2022 and Crack500 enriched the model's training process, improving its generalization across varied road conditions.

The results highlighted the models' abilities to detect, classify, and size pavement cracks accurately, with strong performance in benchmarks when compared to existing approaches. Overall, this chapter underscores the innovative methodologies and their successful application in the domain of pavement crack detection and characterization.

# CHAPTER FIVE

# CONCLUSIONS AND FUTURE WORKS

## 5.1. SUMMARY

This research delves into the critical domain of pavement maintenance, specifically focusing on the detection and characterization of cracks within pavement surfaces, a paramount factor influencing road safety, driving comfort, and infrastructure longevity. This research outlines the imperative of accurate and timely crack detection, shedding light on the limitations of conventional manual inspection methods. These traditional approaches are characterized by their labor-intensive nature, inefficiency, and inherent safety risks, underscoring the urgent need for more sophisticated, reliable, and automated solutions. As vehicular traffic intensifies, the demand for maintaining optimal road conditions escalates, highlighting the necessity for innovative methodologies that can ensure the structural integrity and safety of roadways.

It explores the transformative potential of integrating artificial intelligence, particularly convolutional neural networks (CNNs), into the realm of pavement crack detection. This integration signifies a substantial advancement in the field, offering enhancements in accuracy, speed, and reliability over existing methods. By harnessing the capabilities of deep learning and computer vision, the research aims to transcend the limitations of manual inspections, paving the way for a robust approach to pavement maintenance. This paradigm shift promises not only to refine the process of detecting and characterizing pavement cracks but also to contribute novel insights and methodologies to the fields of civil engineering and infrastructure management, ultimately leading to safer roads and more sustainable infrastructure maintenance practices.

This conclusion section is meticulously organized to align with the research objectives, ensuring a structured and insightful synthesis of the study's findings. This approach not only facilitates a clear and coherent presentation of the outcomes but

also underscores the direct correlation between the set objectives and the resultant achievements. By systematically addressing each objective in its respective conclusion segment, we present a comprehensive overview of the research contributions, thereby enhancing the readability and impact of the conclusions drawn from this study.

Based on the literature review presented in Chapter Two, Objective 1 focuses on evaluating Convolutional Neural Networks (CNNs) for the detection and characterization of pavement cracks. The chapter thoroughly examined various crack types and the associated challenges in accurately detecting them. It also traced the progression from manual inspections to advanced machine learning and deep learning methods, emphasizing their superior accuracy in crack detection. The review detailed the entire process, from data acquisition to pre-processing, and highlighted the significant role of CNN models in improving crack detection capabilities. This analysis supports the selection of appropriate CNN models for pavement crack detection, successfully achieving the goal of Objective 1.

The establishment of an automated pavement crack detection setup, both in lab and field environments, marks a pivotal achievement in enhancing road maintenance analytics. The development of this setup involved a series of steps, starting from the conceptualization in a controlled lab setting to its practical application in a real-world field environment as presented in Chapter 3 of this research. The process included the strategic mounting of a GoPro Hero 8 camera on an inspection vehicle, a methodical approach that ensured the capture of road conditions with exceptional clarity and detail. This setup was not only about installing a camera on a vehicle but also about calibrating it to capture a wide breadth of the road surface, simulating the perspective of road inspectors and maintenance crews.

The calibration process, both in the lab and the field, was critical to ensure that the data collected would be representative of the actual conditions encountered on roads. This included adjusting the camera's height and angle to cover an average lane width, ensuring that the data would be relevant and applicable to a wide range of road types and conditions. The detailed documentation of camera setup specifications and calibration results provided a blueprint for replicating this setup in different contexts, enhancing the scalability of the data collection process. The transition from a lab-

based setup to a field environment was executed with precision, ensuring that the real-world data collection mirrored the controlled conditions of the lab as closely as possible. This seamless integration of lab and field methodologies ensured the reliability and validity of the data collected, providing a strong foundation for the development of deep learning models for pavement crack detection. This approach ensured the capture of high-quality road images under real-world conditions, providing a robust foundation for the database. The preprocessing steps, including image extraction, labeling, augmentation, cropping, etc., further refined the data, ensuring its readiness for deep learning analysis. Objective 2, aimed at developing an automated pavement crack detection database through a vehicle-mounted camera setup to advance road maintenance analytics, was successfully met by assembling a curated dataset.

Objective 3 was successfully met by constructing robust deep learning models designed for the detailed detection and characterization of pavement cracks presented in Chapter 3 of this thesis. This model integrates object detection with precise pixel-level segmentation techniques, leveraging the advanced capabilities of the custom-tailored YOLOv7 and YOLOv8x models. These models were specifically refined to tackle the intricate task of pavement crack analysis, demonstrating outstanding performance in accurately identifying, segmenting, and classifying a wide array of pavement imperfections. A key feature of this framework is the hybrid model, which seamlessly combines the advanced predictive abilities of deep learning with the proven reliability of traditional computational methods. This strategic amalgamation creates a powerful tool that excels in accurately pinpointing and categorizing different types of pavement cracks, thereby enhancing the framework's analytical depth. By offering an in-depth analysis of pavement cracks, including their classifications and dimensions, the framework establishes a new standard in the field of road maintenance. It affords a deeper insight into the condition of road infrastructures, enabling the formulation of more precise and effective maintenance strategies. The achievement of Objective 3 not only highlights the transformative impact of deep learning on enhancing road safety and durability but also paves the way for future advancements in infrastructure care and management.

Objective 4, which focused on evaluating the performance of the developed models for pavement crack detection, has been met with notable success, as evidenced by the analysis presented in Chapter 4 of this thesis. The results from the customized YOLOv7, the precision oriented YOLOv8 X, and the advanced hybrid models demonstrate a high degree of accuracy and reliability in detecting and characterizing pavement cracks. The YOLOv7 model, applied to the Custom dataset and RDD2022 dataset, showcased an overall accuracy of approximately 92%, with precision and recall values highlighting its effectiveness in crack detection. The model's ability to discern between different types of pavement damage—transverse, longitudinal, alligator cracks, and potholes—was particularly noteworthy, with accuracy rates ranging from 85% to 98% across these categories. The YOLOv8 X model, with its advanced segmentation capabilities, further enhanced the framework's precision in identifying cracks. The model demonstrated its proficiency in pavement crack detection through instance segmentation, achieving impressive precision, recall, and accuracy scores of 88%, 90%, and 90% respectively. The utilization of a high-end NVIDIA GeForce RTX 4080 GPU contributed significantly to the efficiency and speed of model training, enabling the handling of complex computations and large datasets. In parallel, our advanced hybrid deep learning model, incorporating Deep Gradient ResNet and a modified attention mechanism, demonstrated robust capabilities across different crack types. For Alligator cracks, the model achieved a precision of 0.84 and a recall of 0.96, highlighting its exceptional ability to detect this crack type accurately. Longitudinal cracks were identified with a precision of 0.89 and a recall of 0.88, while Transverse cracks saw precision and recall rates of 0.87 and 0.80, respectively. These results validate the model's precision in categorizing various crack patterns and its sensitivity in capturing the occurrence of each type.

This research has successfully implemented high-precision measurements of crack lengths at both the pixel level and in actual meters. The capability to measure at the pixel level facilitates detailed analytical assessments, while meter-level measurements are indispensable for practical applications, including maintenance planning and risk management. The integration of rigorous calibration processes and laboratory validations ensures that these measurements are both accurate and consistent across varying conditions and setups, thereby enhancing the reliability and applicability of the research in real-world scenarios. Moreover, this study introduces

an effective method for concatenating large cracks across multiple frames, addressing a prevalent challenge in large-scale infrastructure evaluations. Traditional assessments often struggle with cracks that span beyond a single frame. The developed method connects these extended crack patterns, providing a holistic view of infrastructure conditions, enabling the model to effectively perform detection, classification, and characterization processes. Furthermore, comprehensive benchmarking against established standards highlights the models' capabilities in this domain. Their success in not only detecting and classifying various crack types but also in accurately measuring these cracks in tangible real-world metrics signifies the fulfillment of Objective 4, demonstrating a significant advancement in the field of pavement crack analysis.

## 5.2. FUTURE WORKS

While this study has made significant strides in the field of pavement crack detection and analysis, there remain several avenues for further exploration and enhancement. The following section outlines potential future works that can build upon the foundations laid in this research, addressing existing limitations and advancing the state-of-the-art in pavement monitoring and maintenance. By delving into these areas of inquiry, researchers and practitioners can continue to innovate and refine methodologies for more accurate, efficient, and proactive management of infrastructure assets. From leveraging emerging technologies to exploring novel analytical approaches, the proposed future works aim to contribute to the ongoing evolution of pavement management systems, ultimately promoting safer and more sustainable transportation networks.

1. **Integration of Multimodal Data:** Explore the integration of various data sources such as high-resolution imagery, LiDAR scans, and pavement surface temperature data to enhance crack detection accuracy and robustness. By combining different modalities, the models can capture a more comprehensive understanding of pavement conditions, leading to improved performance.
2. **Transfer Learning for Domain Adaptation**: Explore the application of transfer learning techniques to adapt pre-trained crack detection models to

different geographic regions or pavement surface materials. By fine-tuning the models on target domain data, future researchers can mitigate the need for extensive labelled data collection and achieve better generalization performance across diverse environments.

3. **Real-time Crack Monitoring Systems:** Develop real-time crack monitoring systems using edge computing and Internet of Things (IoT) devices installed on vehicles or drones. These systems can continuously capture pavement images and analyze them on the fly to provide timely alerts for maintenance interventions, helping to prevent further deterioration and ensure road safety.
4. **Longitudinal Analysis for Predictive Maintenance:** Conduct longitudinal studies to analyze the evolution of pavement cracks over time and develop predictive maintenance models. By leveraging historical crack data and environmental factors, such as weather and traffic load, these models can forecast future pavement degradation trends and optimize maintenance schedules for cost-effective infrastructure management.
5. **Human-in-the-Loop Approaches for Model Interpretability:** Explore human-in-the-loop approaches to enhance the interpretability of crack detection models and facilitate user trust and collaboration. By integrating human feedback mechanisms into the model training and inference pipeline, researchers can improve transparency, validate model predictions, and refine algorithmic performance based on domain experts' insights.

# REFERENCES


Abdulateef, S., & Salman, M. (2021). A Comprehensive Review of Image Segmentation Techniques. *Iraqi Journal for Electrical and Electronic Engineering*, *17*(2), 166–175. https://doi.org/10.37917/ijeee.17.2.18

Abu-Ain, W., Abdullah, S. N. H. S., Bataineh, B., Abu-Ain, T., & Omar, K. (2013). Skeletonization Algorithm for Binary Images. *Procedia Technology*, *11*, 704–709. https://doi.org/10.1016/j.protcy.2013.12.248

Akagic, A., Buza, E., Omanovic, S., & Karabegovic, A. (2018). Pavement crack detection using Otsu thresholding for image segmentation. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 1092–1097. https://doi.org/10.23919/MIPRO.2018.8400199

Akarsu, B., KARAKÖSE, M., PARLAK, K., AKIN, E., & SARIMADEN, A. (2016). A Fast and Adaptive Road Defect Detection Approach Using Computer Vision with Real Time Implementation. *International Journal of Applied Mathematics, Electronics and Computers*, 290–290. https://doi.org/10.18100/ijamec.270546

Alshandah, M., Huang, Y., Gao, Z., & Lu, P. (2020). Internal crack detection in concrete pavement using discrete strain sensors. *Journal of Civil Structural Health Monitoring*, *10*(2), 345–356. https://doi.org/10.1007/s13349-020-00388-2

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, *8*(1), 53. https://doi.org/10.1186/s40537-021-00444-8

Amhaz, R., Chambon, S., Idier, J., & Baltazart, V. (2016). Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Transactions on Intelligent Transportation Systems*, *17*(10), 2718–2729. https://doi.org/10.1109/TITS.2015.2477675

Arbelaez, P., Pont-Tuset, J., Barron, J., Marques, F., & Malik, J. (2014). Multiscale Combinatorial Grouping. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 328–335. https://doi.org/10.1109/CVPR.2014.49

Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., & Sekimoto, Y. (2021). RDD2020: An annotated image dataset for automatic road damage detection using deep learning. *Data in Brief*, *36*, 107133. https://doi.org/10.1016/j.dib.2021.107133

Arya, D., Maeda, H., Ghosh, S. K., Toshniwal, D., & Sekimoto, Y. (2022). *RDD2022: A multi-national image dataset for automatic Road Damage Detection*.

Ashraf, A., Gunawan, T. S., Rahman, F. D. A., & Kartiwi, M. (2022). *A Summarization of Image and Video Databases for Emotion Recognition* (pp. 669–680). https://doi.org/10.1007/978-981-33-4597-3_60

Ashraf, A., Gunawan, T. S., Rahman, F. D. A., Sophian, A., Ambikairajah, E., Ihsanto, E., & Kartiwi, M. (2021). *Affective Computing for Visual Emotion Recognition Using Convolutional Neural Networks* (pp. 11–20). https://doi.org/10.1007/978-3-030-70917-4_2

Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., & Ismail, N. N. (2023). Machine learning-based pavement crack detection, classification, and characterization: a review. *Bulletin of Electrical Engineering and Informatics*, *12*(6), 3601–3619. https://doi.org/10.11591/eei.v12i6.5345

Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., Ismail, N. N., & Bawono, A. A. (2022a). Detection of Road Cracks Using Convolutional Neural Networks and Threshold Segmentation. *Journal of Integrated and Advanced Engineering (JIAE)*, *2*(2), 123–134. https://doi.org/10.51662/jiae.v2i2.82

Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., Ismail, N. N., & Bawono, A. A. (2022b). Detection of Road Cracks Using Convolutional Neural Networks and Threshold Segmentation. *Journal of Integrated and Advanced Engineering (JIAE)*, *2*(2), 123–134. https://doi.org/10.51662/jiae.v2i2.82

Ashraf, A., Surya Gunawan, T., Subhan Riza, B., Haryanto, E. V., & Janin, Z. (2020). On the review of image and video-based depression detection using machine

learning. *Indonesian Journal of Electrical Engineering and Computer Science*, *19*(3), 1677. https://doi.org/10.11591/ijeecs.v19.i3.pp1677-1684

B. T. Passos, M. J. Cassaniga, A. M. da R. Fernandes, K. B. Medeiros, & E. Comunello. (2020). Cracks and potholes in road images. *Mendeley Data*, *3*.

Berahmand, K., Daneshfar, F., Salehi, E. S., Li, Y., & Xu, Y. (2024). Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, *57*(2), 28. https://doi.org/10.1007/s10462-023-10662-6

Bhat, S., Naik, S., Gaonkar, M., Sawant, P., Aswale, S., & Shetgaonkar, P. (2021). Road crack detection using convolutional neural network. *Indian Journal of Science and Technology*, *14*(10), 881–891. https://doi.org/10.17485/IJST/v14i10.1245

Bi, Z., & Cao, P. (2021). Color space conversion algorithm and comparison study. *Journal of Physics: Conference Series*, *1976*(1), 012008. https://doi.org/10.1088/1742-6596/1976/1/012008

Cao, W., Yuan, J., He, Z., Zhang, Z., & He, Z. (2018). Fast Deep Neural Networks With Knowledge Guided Training and Predicted Regions of Interests for Real-Time Video Object Detection. *IEEE Access*, *6*, 8990–8999. https://doi.org/10.1109/ACCESS.2018.2795798

Chambon, S., & Moliard, J.-M. (2011). Automatic Road Pavement Assessment with Image Processing: Review and Comparison. *International Journal of Geophysics*, *2011*, 1–20. https://doi.org/10.1155/2011/989354

Chen, H., Su, Y., & He, W. (2021). Automatic crack segmentation using deep high-resolution representation learning. *Applied Optics*, *60*(21), 6080. https://doi.org/10.1364/AO.423406

CHUN, P., HASHIMOTO, K., KATAOKA, N., KURAMOTO, N., & OHGA, M. (2015). ASPHALT PAVEMENT CRACK DETECTION USING IMAGE PROCESSING AND NAÏVE BAYES BASED MACHINE LEARNING APPROACH. *Journal of Japan Society of Civil Engineers, Ser. E1 (Pavement Engineering)*, *70*(3), I_1-I_8. https://doi.org/10.2208/jscejpe.70.I_1

Chun, P., Yamane, T., & Tsuzuki, Y. (2021). Automatic Detection of Cracks in Asphalt Pavement Using Deep Learning to Overcome Weaknesses in Images and GIS Visualization. *Applied Sciences*, *11*(3), 892. https://doi.org/10.3390/app11030892

Dadrasjavan, F., Zarrinpanjeh, N., & Ameri, A. (2019). Automatic Crack Detection of Road Pavement Based on Aerial UAV Imagery. *Preprints* , 1–16.

Dalal, N., & Triggs, B. (n.d.). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Decker, D. S. (2016, June 30). Best practices for crack treatments in asphalt pavements. *Proceedings of 6th Eurasphalt & Eurobitume Congress*. https://doi.org/10.14311/EE.2016.045

Dong, X. L., & Rekatsinas, T. (2018). Data Integration and Machine Learning. *Proceedings of the 2018 International Conference on Management of Data*, 1645–1650. https://doi.org/10.1145/3183713.3197387

Dung, C. V., & Anh, L. D. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, *99*, 52–58. https://doi.org/10.1016/j.autcon.2018.11.028

Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U., & Gross, H.-M. (2017). How to get pavement distress detection ready for deep learning? A systematic approach. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2039–2047. https://doi.org/10.1109/IJCNN.2017.7966101

Fan, Z., & Hu, J. (2019). Review and Prospect of Research on Generative Adversarial Networks. *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, 726–730. https://doi.org/10.1109/ICCSN.2019.8905263

Fan, Z., Li, C., Chen, Y., Wei, J., Loprencipe, G., Chen, X., & Di Mascio, P. (2020). Automatic Crack Detection on Road Pavements Using Encoder-Decoder Architecture. *Materials*, *13*(13), 2960. https://doi.org/10.3390/ma13132960

Fan, Z., Wu, Y., Lu, J., & Li, W. (2018a). *Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network*.

Fan, Z., Wu, Y., Lu, J., & Li, W. (2018b). *Automatic Pavement Crack Detection Based on Structured Prediction with the Convolutional Neural Network*.

Fang, L., Shi, Z., Liu, Y., Li, C., Pang, M., & Zhao, E. (2023). A general geometric transformation model for line-scan image registration. *EURASIP Journal on Advances in Signal Processing*, *2023*(1), 78. https://doi.org/10.1186/s13634-023-01041-y

Feng, X., Xiao, L., Li, W., Pei, L., Sun, Z., Ma, Z., Shen, H., & Ju, H. (2020). Pavement Crack Detection and Segmentation Method Based on Improved Deep Learning Fusion Model. *Mathematical Problems in Engineering*, *2020*, 1–22. https://doi.org/10.1155/2020/8515213

Ganesan, P., & Sajiv, G. (2017). A comprehensive study of edge detection for image processing applications. *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, 1–6. https://doi.org/10.1109/ICIIECS.2017.8275968

Gawhade, R., Bohara, L. R., Mathew, J., & Bari, P. (2022). Computerized Data-Preprocessing To Improve Data Quality. *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, 1–6. https://doi.org/10.1109/ICPC2T53885.2022.9776676

Gehri, N., Mata-Falcón, J., & Kaufmann, W. (2020). Automated crack detection and measurement based on digital image correlation. *Construction and Building Materials*, *256*, 119383. https://doi.org/10.1016/j.conbuildmat.2020.119383

Ghimire, A., Thapa, S., Jha, A. K., Adhikari, S., & Kumar, A. (2020). Accelerating Business Growth with Big Data and Artificial Intelligence. *2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 441–448. https://doi.org/10.1109/I-SMAC49090.2020.9243318

Ghimire, A., Thapa, S., Jha, A. K., Kumar, A., Kumar, A., & Adhikari, S. (2020). AI and IoT Solutions for Tackling COVID-19 Pandemic. *2020 4th International*

*Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 1083–1092. https://doi.org/10.1109/ICECA49313.2020.9297454

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Gunawan, T. S., Alghifari, M. F., Morshidi, M. A., & Kartiwi, M. (2018). A Review on Emotion Recognition Algorithms using Speech Analysis. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, *6*(1). https://doi.org/10.52549/ijeei.v6i1.409

Ha, J., Kim, D., & Kim, M. (2022). Assessing severity of road cracks using deep learning-based segmentation and detection. *The Journal of Supercomputing*, *78*(16), 17721–17735. https://doi.org/10.1007/s11227-022-04560-x

Han, T., Jiang, D., Zhao, Q., Wang, L., & Yin, K. (2018). Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*, *40*(8), 2681–2693. https://doi.org/10.1177/0142331217708242

He, K., Zhang, X., Ren, S., & Sun, J. (2015a). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://api.semanticscholar.org/CorpusID:206594692

He, K., Zhang, X., Ren, S., & Sun, J. (2015b). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

Hu, G. X., Hu, B. L., Yang, Z., Huang, L., & Li, P. (2021). Pavement Crack Detection Method Based on Deep Learning Models. *Wireless Communications and Mobile Computing*, *2021*, 1–13. https://doi.org/10.1155/2021/5573590

Humeau-Heurtier, A. (2022). Color Texture Analysis: A Survey. *IEEE Access*, *10*, 107993–108003. https://doi.org/10.1109/ACCESS.2022.3213439

Ihsanto, E., Ramli, K., Sudiana, D., & Gunawan, T. S. (2020). Fast and Accurate Algorithm for ECG Authentication Using Residual Depthwise Separable Convolutional Neural Networks. *Applied Sciences*, *10*(9), 3304. https://doi.org/10.3390/app10093304

Jahangiri, A., & Rakha, H. A. (2015). Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2406–2417. https://doi.org/10.1109/TITS.2015.2405759

Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access*, *7*, 133653–133667. https://doi.org/10.1109/ACCESS.2019.2941229

Jayanth Balaji, A., Thiru Balaji, G., Dinesh, M. S., Nair, B., & D. S, H. R. (2018). Asphalt Crack Dataset. *Mendeley Data*,.

Jin, H., Wan, F., & Ruan, O. (2018). *Pavement Crack Detection Fused HOG and Watershed Algorithm of Range Image* (pp. 475–488). https://doi.org/10.1007/978-3-319-59463-7_47

Jo, Y., & Ryu, S. (2015). Pothole Detection System Using a Black-box Camera. *Sensors*, *15*(11), 29316–29331. https://doi.org/10.3390/s151129316

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8*.

Kamdi, S., & Krishna, R. (2012). *Image Segmentation and Region Growing Algorithm*. https://api.semanticscholar.org/CorpusID:16748401

Kar, M. K., Nath, M. K., & Neog, D. R. (2021). A Review on Progress in Semantic Image Segmentation and Its Application to Medical Images. *SN Computer Science*, *2*(5), 397. https://doi.org/10.1007/s42979-021-00784-5

Koch, C., Georgieva, K., Kasireddy, V., Akinci, B., & Fieguth, P. (2015). A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Advanced Engineering Informatics*, *29*(2), 196–210. https://doi.org/10.1016/j.aei.2015.01.008

Kumar, G., & Bhatia, P. K. (2014). A Detailed Review of Feature Extraction in Image Processing Systems. *2014 Fourth International Conference on Advanced*

*Computing & Communication Technologies*, 5–12. https://doi.org/10.1109/ACCT.2014.74

Li, G., Chen, Y., Zhou, J., Zheng, X., & Li, X. (2022). Road crack detection and quantification based on segmentation network using architecture of matrix. *Engineering Computations*, *39*(2), 693–721. https://doi.org/10.1108/EC-01-2021-0043

Li, H., Song, D., Liu, Y., & Li, B. (2019). Automatic Pavement Crack Detection by Multi-Scale Image Fusion. *IEEE Transactions on Intelligent Transportation Systems*, *20*(6), 2025–2036. https://doi.org/10.1109/TITS.2018.2856928

Li, P., Xia, H., Zhou, B., Yan, F., & Guo, R. (2022a). A Method to Improve the Accuracy of Pavement Crack Identification by Combining a Semantic Segmentation and Edge Detection Model. *Applied Sciences*, *12*(9), 4714. https://doi.org/10.3390/app12094714

Li, P., Xia, H., Zhou, B., Yan, F., & Guo, R. (2022b). A Method to Improve the Accuracy of Pavement Crack Identification by Combining a Semantic Segmentation and Edge Detection Model. *Applied Sciences*, *12*(9), 4714. https://doi.org/10.3390/app12094714

Li, X., & Chen, D. (2022). A survey on deep learning-based panoptic segmentation. *Digital Signal Processing*, *120*, 103283. https://doi.org/10.1016/j.dsp.2021.103283

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*, *3*, 111–132. https://doi.org/10.1016/j.aiopen.2022.10.001

Liong, S.-T., Gan, Y. S., Huang, Y.-C., Yuan, C.-A., & Chang, H.-C. (2019). *Automatic Defect Segmentation on Leather with Deep Learning*.

Liu, J., Yang*, X., & Lee, V. C. S. (2020). Automated pavement crack detection using region-based convolutional neural network. In *Functional Pavements* (pp. 248–252). CRC Press. https://doi.org/10.1201/9781003156222-38

Liu, W., Huang, Y., Li, Y., & Chen, Q. (2019). *FPCNet: Fast Pavement Crack Detection Network Based on Encoder-Decoder Architecture*.

Lozano-Vázquez, L. V., Miura, J., Rosales-Silva, A. J., Luviano-Juárez, A., & Mújica-Vargas, D. (2022). Analysis of Different Image Enhancement and Feature Extraction Methods. *Mathematics*, *10*(14), 2407. https://doi.org/10.3390/math10142407

Ma, D., Fang, H., Xue, B., Wang, F., A. Msekh, M., & Ling Chan, C. (2020). Intelligent Detection Model Based on a Fully Convolutional Neural Network for Pavement Cracks. *Computer Modeling in Engineering & Sciences*, *123*(3), 1267–1291. https://doi.org/10.32604/cmes.2020.09122

Macukow, B. (2016). *Neural Networks – State of Art, Brief History, Basic Models and Architecture* (pp. 3–14). https://doi.org/10.1007/978-3-319-45378-1_1

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). *Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone*. https://doi.org/10.1111/mice.12387

Mahadevkar, S. V., Khemani, B., Patil, S., Kotecha, K., Vora, D. R., Abraham, A., & Gabralla, L. A. (2022). A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions. *IEEE Access*, *10*, 107293–107329. https://doi.org/10.1109/ACCESS.2022.3209825

Mei, Q., Gül, M., & Azim, M. R. (2020). Densely connected deep neural network considering connectivity of pixels for automatic crack detection. *Automation in Construction*, *110*, 103018. https://doi.org/10.1016/j.autcon.2019.103018

Mo, Y., Wu, Y., Yang, X., Liu, F., & Liao, Y. (2022). Review the state-of-the-art technologies of semantic segmentation based on deep learning. *Neurocomputing*, *493*, 626–646. https://doi.org/10.1016/j.neucom.2022.01.005

Mohan, A., & Poobal, S. (2018a). Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, *57*(2), 787–798. https://doi.org/10.1016/j.aej.2017.01.020

Mohan, A., & Poobal, S. (2018b). Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, *57*(2), 787–798. https://doi.org/10.1016/j.aej.2017.01.020

Muhammad Ali, P., & Faraj, R. (2014). *Data Normalization and Standardization: A Technical Report*. https://doi.org/10.13140/RG.2.2.28948.04489

Munawar, H. S., Hammad, A. W. A., Haddad, A., Soares, C. A. P., & Waller, S. T. (2021). Image-Based Crack Detection Methods: A Review. *Infrastructures*, *6*(8), 115. https://doi.org/10.3390/infrastructures6080115

Mustafa, W. A., & Abdul Kader, M. M. M. (2018). A Review of Histogram Equalization Techniques in Image Enhancement Application. *Journal of Physics: Conference Series*, *1019*, 012026. https://doi.org/10.1088/1742-6596/1019/1/012026

Naik, S. K., & Murthy, C. A. (2003). Hue-preserving color image enhancement without gamut problem. *IEEE Transactions on Image Processing*, *12*(12), 1591–1598. https://doi.org/10.1109/TIP.2003.819231

Ni, F., Zhang, J., & Chen, Z. (2019). Pixel-level crack delineation in images with convolutional feature fusion. *Structural Control and Health Monitoring*, *26*(1), e2286. https://doi.org/10.1002/stc.2286

Nie, M., & Wang, C. (2019). Pavement Crack Detection based on yolo v3. *2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, 327–330. https://doi.org/10.1109/IICSPI48186.2019.9095956

Oliveira, H., & Correia, P. L. (2013). Automatic Road Crack Detection and Characterization. *IEEE Transactions on Intelligent Transportation Systems*, *14*(1), 155–168. https://doi.org/10.1109/TITS.2012.2208630

Olson, M., Wyner, A., & Berk, R. (2018). Modern Neural Networks Generalize on Small Data Sets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/fface8385abbf94b4593a0ed53a0c70f-Paper.pdf

Özgenel, Ç. F., & Sorguç, A. G. (2018, July 22). *Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings*. https://doi.org/10.22260/ISARC2018/0094

Pantuso, A., Loprencipe, G., Bonin, G., & Teltayev, B. B. (2019). Analysis of Pavement Condition Survey Data for Effective Implementation of a Network Level Pavement Management Program for Kazakhstan. *Sustainability*, *11*(3), 901. https://doi.org/10.3390/su11030901

Peng, L., Chao, W., Shuangmiao, L., & Baocai, F. (2015). Research on Crack Detection Method of Airport Runway Based on Twice-Threshold Segmentation. *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, 1716–1720. https://doi.org/10.1109/IMCCC.2015.364

Ranyal, E., Sadhu, A., & Jain, K. (2024). Enhancing pavement health assessment: An attention-based approach for accurate crack detection, measurement, and mapping. *Expert Systems with Applications*, *247*, 123314. https://doi.org/10.1016/j.eswa.2024.123314

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*.

Rokach, L., & Maimon, O. (n.d.). Decision Trees. In *Data Mining and Knowledge Discovery Handbook* (pp. 165–192). Springer-Verlag. https://doi.org/10.1007/0-387-25465-X_9

Said, K. A. M., & Jambek, A. B. (2021). Analysis of Image Processing Using Morphological Erosion and Dilation. *Journal of Physics: Conference Series*, *2071*(1), 012033. https://doi.org/10.1088/1742-6596/2071/1/012033

Sari, Y., Prakoso, P. B., & Baskara, A. R. (2019). Road Crack Detection using Support Vector Machine (SVM) and OTSU Algorithm. *2019 6th International Conference on Electric Vehicular Technology (ICEVT)*, 349–354. https://doi.org/10.1109/ICEVT48285.2019.8993969

Sarker, I. H. (2021). Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. *SN Computer Science*, *2*(6), 420. https://doi.org/10.1007/s42979-021-00815-1

Schneider, A., Hommel, G., & Blettner, M. (2010). Linear Regression Analysis. *Deutsches Ärzteblatt International*. https://doi.org/10.3238/arztebl.2010.0776

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *CoRR, abs/1312.6229*. https://api.semanticscholar.org/CorpusID:4071727

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*, *404*, 132306. https://doi.org/10.1016/j.physd.2019.132306

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016a). Automatic Road Crack Detection Using Random Structured Forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434–3445. https://doi.org/10.1109/TITS.2016.2552248

Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z. (2016b). Automatic Road Crack Detection Using Random Structured Forests. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3434–3445. https://doi.org/10.1109/TITS.2016.2552248

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Sivamayil, K., Rajasekar, E., Aljafari, B., Nikolovski, S., Vairavasundaram, S., & Vairavasundaram, I. (2023). A Systematic Study on Reinforcement Learning Based Applications. *Energies*, *16*(3), 1512. https://doi.org/10.3390/en16031512

Talab, A. M. A., Huang, Z., Xi, F., & HaiMing, L. (2016). Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik*, *127*(3), 1030–1033. https://doi.org/10.1016/j.ijleo.2015.09.147

Tran, V. P., Tran, T. S., Lee, H. J., Kim, K. D., Baek, J., & Nguyen, T. T. (2021). One stage detector (RetinaNet)-based crack detection for asphalt pavements considering pavement distresses and surface objects. *Journal of Civil Structural Health Monitoring*, *11*(1), 205–222. https://doi.org/10.1007/s13349-020-00447-8

Uzakkyzy, N., Ismailova, A., Ayazbaev, T., Beldeubayeva, Z., Kodanova, S., Utenova, B., Satybaldiyeva, A., & Kaldarova, M. (2023). Image noise reduction

by deep learning methods. *International Journal of Electrical and Computer Engineering (IJECE)*, *13*(6), 6855. https://doi.org/10.11591/ijece.v13i6.pp6855-6861

Viola, P., & Jones, M. (n.d.). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, I-511-I–518. https://doi.org/10.1109/CVPR.2001.990517

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*.

Wang, X., Zhaozheng, H., Li, N., & Qin, L. (2020). Pavement crack analysis by referring to historical crack data based on multi-scale localization. *PLOS ONE*, *15*(8), e0235171. https://doi.org/10.1371/journal.pone.0235171

Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X., & Ling, H. (2020). Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection. *IEEE Transactions on Intelligent Transportation Systems*, *21*(4), 1525–1535. https://doi.org/10.1109/TITS.2019.2910595

Yang, X., Li, H., Yu, Y., Luo, X., Huang, T., & Yang, X. (2018). Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), 1090–1109. https://doi.org/10.1111/mice.12412

Yu, Y., Wang, C., Fu, Q., Kou, R., Huang, F., Yang, B., Yang, T., & Gao, M. (2023). Techniques and Challenges of Image Segmentation: A Review. *Electronics*, *12*(5), 1199. https://doi.org/10.3390/electronics12051199

Zalama, E., Gómez-García-Bermejo, J., Medina, R., & Llamas, J. (2014). Road Crack Detection Using Visual Features Extracted by Gabor Filters. *Computer-Aided Civil and Infrastructure Engineering*, *29*(5), 342–358. https://doi.org/10.1111/mice.12042

Zeger, I., & Grgic, S. (2020). An Overview of Grayscale Image Colorization Methods. *2020 International Symposium ELMAR*, 109–112. https://doi.org/10.1109/ELMAR49956.2020.9219019

Zeiada, W., Hamad, K., Omar, M., Underwood, B. S., Khalil, M. A., & Karzad, A. S. (2019). Investigation and modelling of asphalt pavement performance in cold regions. *International Journal of Pavement Engineering*, *20*(8), 986–997. https://doi.org/10.1080/10298436.2017.1373391

Zhang, A., Wang, K. C. P., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J. Q., & Chen, C. (2017). Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network. *Computer-Aided Civil and Infrastructure Engineering*, *32*(10), 805–819. https://doi.org/10.1111/mice.12297

Zhang, L., Yang, F., Daniel Zhang, Y., & Zhu, Y. J. (2016). Road crack detection using deep convolutional neural network. *2016 IEEE International Conference on Image Processing (ICIP)*, 3708–3712. https://doi.org/10.1109/ICIP.2016.7533052

Zhao, S., & H. Ning. (2017). Detection of pavement cracks based on convolutional neural network. *Sensors and Microsystems*, *36*(11), 135–138.

Zhao, Y., Zhou, L., Wang, X., Wang, F., & Shi, G. (2023a). Highway Crack Detection and Classification Using UAV Remote Sensing Images Based on CrackNet and CrackClassification. *Applied Sciences*, *13*(12), 7269. https://doi.org/10.3390/app13127269

Zhao, Y., Zhou, L., Wang, X., Wang, F., & Shi, G. (2023b). Highway Crack Detection and Classification Using UAV Remote Sensing Images Based on CrackNet and CrackClassification. *Applied Sciences*, *13*(12), 7269. https://doi.org/10.3390/app13127269

Zhou, S., Liang, Y., Wan, J., & Li, S. Z. (2016). *Facial Expression Recognition Based on Multi-scale CNNs* (pp. 503–510). https://doi.org/10.1007/978-3-319-46654-5_55

Zhou, Y., Wang, F., Meghanathan, N., & Huang, Y. (2016). Seed-Based Approach for Automated Crack Detection from Pavement Images. *Transportation Research Record: Journal of the Transportation Research Board*, *2589*(1), 162–171. https://doi.org/10.3141/2589-18

Zitnick, C. L., & Dollár, P. (2014). *Edge Boxes: Locating Object Proposals from Edges* (pp. 391–405). https://doi.org/10.1007/978-3-319-10602-1_26

Zou, Q., Zhang, Z., Li, Q., Qi, X., Wang, Q., & Wang, S. (2019). DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection. *IEEE Transactions on Image Processing*, *28*(3), 1498–1512. https://doi.org/10.1109/TIP.2018.2878966

Zou, X., Hu, Y., Tian, Z., & Shen, K. (2019). Logistic Regression Model Optimization and Case Analysis. *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, 135–139. https://doi.org/10.1109/ICCSNT47585.2019.8962457

# LIST OF PUBLICATIONS

1. Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., & Ismail, N. N. (2023). Machine learning-based pavement crack detection, classification, and characterization: a review. Bulletin of Electrical Engineering and Informatics, 12(6), 3601-3619.
2. Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., Ismail, N. N., & Bawono, A. A. (2023). Efficient Pavement Crack Detection and Classification Using Custom YOLOv7 Model. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 11(1), 119-132.
3. Ashraf, A., Sophian, A., Shafie, A. A., Gunawan, T. S., Ismail, N. N., & Bawono, A. A. (2022). Detection of Road Cracks Using Convolutional Neural Networks and Threshold Segmentation. Journal of Integrated and Advanced Engineering (JIAE), 2(2), 123-134.
4. Yusof, N. I. M., Sophian, A., Zaki, H. F. M., Bawono, A. A., & Ashraf, A. (2024). Assessing the performance of YOLOv5, YOLOv6, and YOLOv7 in road defect detection and classification: a comparative study. Bulletin of Electrical Engineering and Informatics, 13(1), 350-360.

# APPENDIX I

## ACHIEVEMENT OUTCOMES

1. Awarded Silver by Kulliyyah of Engineering for the research on “Pavement Crack Detection and Characterization Using Deep Learning and Pixel Level Segmentation”, presented at Kulliyyah of Engineering Research, Innovation and Commercialization Exhibition (KERICE) 2024.
2. Awarded Second Place in the Research Video Competition at the Kulliyyah of Engineering, International Islamic University Malaysia for the research on “Pavement Crack Detection and Characterization Using Deep Learning and Pixel Level Segmentation”.
3. Development of a road crack dataset named RCD-IIUM: Captured video frames from Kuala Lumpur and Selangor regions were methodically extracted and annotated to create a comprehensive real-world data repository.
4. Awarded Third Place in the Three Minute Thesis Competition at the Kulliyyah of Engineering, International Islamic University Malaysia.

# APPENDIX II

## CODES

### Segmentation

```
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.utils import normalize
import tensorflow as tf
import os
import glob
import cv2
from matplotlib import pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping,
ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras import backend as K
from statistics import mean
from keras.models import load_model
import random
from pycm import *
from sklearn.metrics import multilabel_confusion_matrix
from sklearn import metrics

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors
import imageio
import scipy, scipy.misc, scipy.signal
import cv2
import sys
from tkinter import Tcl
"""PRE PROCESSING"""

"""FOR ORIGINAL IMAGE"""

#Resizing images
SIZE_X = 128
SIZE_Y = 128

#Number of classes for segmentation
n_classes=2
file_list = os.listdir("Original/")
kk = Tcl().call('lsort', '-dict', file_list)

names = list(kk)
```

```
full_images = []

for directory_path in names:
      img = cv2.imread(f"Original/{directory_path}",0)
      img = cv2.resize(img, (SIZE_Y, SIZE_X))
      full_images.append(img)

# Convert list to array
full_images = np.array(full_images)
"""FOR GROUND TRUTH"""

# Get the List from the Images
file_list_label = os.listdir("Mask/")

kk_label = Tcl().call('lsort', '-dict', file_list_label)

names_label = list(kk_label)


#Mask Images
train_masks = []
for directory_path in names_label:
      mask = cv2.imread(f"Mask/{directory_path}",0)
      mask = cv2.resize(mask, (SIZE_Y, SIZE_X), interpolation =
cv2.INTER_NEAREST)
      train_masks.append(mask)

#Convert list to array
train_masks = np.array(train_masks)
train_masks[train_masks > 0] = 255
np.unique(train_masks)
# Drop the last image
lastElementIndex = len(full_images)-1
full_images = full_images[:lastElementIndex]
full_images.shape
# Displaying the Train Image
ll = 3371
plt.imshow(full_images[ll], cmap='gray')
plt.show()
plt.imshow(train_masks[ll], cmap='gray')
plt.show()
"""### LABEL ENCODING THE MASK FOR THE MULTI CLASS SEGMENTATION"""

from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()

n, h, w  = train_masks.shape
train_masks_reshaped = train_masks.reshape(-1,1)
train_masks_reshaped_encoded =
```

```
labelencoder.fit_transform(train_masks_reshaped)
train_masks_encoded_original_shape =
train_masks_reshaped_encoded.reshape(n, h, w)

np.unique(train_masks_encoded_original_shape)

full_images.shape

train_masks_input =
np.expand_dims(train_masks_encoded_original_shape, axis=3)
print("Image shape",full_images.shape)
print("Mask shape",train_masks_input.shape)
#-----------------------------------------------------------------
----------------
"""### TRAIN TEST SPLIT"""

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(full_images,
train_masks_input, test_size = 0.10)
print("Class values in the dataset are ... ", np.unique(y_train))

"""CONVERTING INTO CATEGORICAL"""

from tensorflow.keras.utils import to_categorical
train_masks_cat = to_categorical(y_train, num_classes=n_classes)
y_train_cat = train_masks_cat.reshape((y_train.shape[0],
y_train.shape[1], y_train.shape[2], n_classes))

test_masks_cat = to_categorical(y_test, num_classes=n_classes)
y_test_cat = test_masks_cat.reshape((y_test.shape[0],
y_test.shape[1], y_test.shape[2], n_classes))
print("Class values in the dataset are ... ", np.unique(y_train))
#-----------------------------------------------------------------
----------------
from tensorflow.keras.metrics import Precision, Recall,
MeanIoU,Accuracy
import torch
"""DECLARING PERFORMANCE METRICS"""

smooth = 1.
def dice_coef(y_true, y_pred):
    y_true_f = tf.keras.layers.Flatten()(y_true)
    y_pred_f = tf.keras.layers.Flatten()(y_pred)
    intersection = tf.reduce_sum(y_true_f * y_pred_f)
    return (2. * intersection + smooth) / (tf.reduce_sum(y_true_f)
+ tf.reduce_sum(y_pred_f) + smooth)

def accuracy(y_true, y_pred):
```

```
    return K.mean(K.equal(y_true, K.round(y_pred)), axis=-1)

def jacard_coef(y_true, y_pred):
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)
    intersection = K.sum(y_true_f * y_pred_f)
    return (intersection + 1.0) / (K.sum(y_true_f) +
K.sum(y_pred_f) - intersection + 1.0)

def dice_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred)

metrics = [Recall(), Precision(), dice_coef,accuracy,jacard_coef]


 IMPORT LIBRARIES
from keras.models import Model
from keras.layers import Input, Conv2D, MaxPooling2D,
UpSampling2D, concatenate, Conv2DTranspose, BatchNormalization,
Dropout, Lambda
import os
import numpy as np
import cv2

import tensorflow as tf
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model

#-------------------------------------------------------------------
-------------------------------------------------------------------
------------------------------
"""OPTIMIZED CONVOLUTIONAL BLOCK"""

#from keras.engine import InputSpec
from keras import backend as K
from keras import initializers
from keras import regularizers
from keras import constraints
from tensorflow import keras
from tensorflow.keras.layers import Layer

class AugumentedSubPixelShuffling(Layer):

    def __init__(self, step_dim,
                 W_regularizer=None, b_regularizer=None,
                 W_constraint=None, b_constraint=None,
                 bias=True, **kwargs):

        self.bias = bias
```

```
        self.step_dim = step_dim
        self.features_dim = 0

        self.supports_masking = True
        self.init = keras.initializers.get('glorot_uniform')

        self.W_regularizer = keras.regularizers.get(W_regularizer)
        self.b_regularizer = keras.regularizers.get(b_regularizer)

        self.W_constraint = keras.constraints.get(W_constraint)
        self.b_constraint = keras.constraints.get(b_constraint)

        super(AugumentedSubPixelShuffling,
self).__init__(**kwargs)


    def weight_adding(self, input_shape):
        assert len(input_shape) == 3

        self.W = self.add_weight(shape=(input_shape[-1],),
                                 initializer=self.init,
                                 name='{}_W'.format(self.name),
                                 regularizer=self.W_regularizer,
                                 constraint=self.W_constraint)
        self.features_dim = input_shape[-1]

        if self.bias:
            self.b = self.add_weight(shape=(input_shape[1],),
                                     initializer='zero',

 name='{}_b'.format(self.name),

 regularizer=self.b_regularizer,
                                     constraint=self.b_constraint)
        else:
            self.b = None

        self.built = True

class Deep_Gradient_ResNet:
    def __init__(self, input_size=256,n_classes=2):
        self.input_size = input_size
        self.n_classes = n_classes

# CNN Enchanced Architecture
    def build_upsampling_block(self):
        def conv_block(x, n_filter):
            x_init = x
```

```
            ## Conv 1
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)

            ## Conv 2
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (3, 3), padding="same")(x)

            ## Conv 3
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)
            ## Conv 4
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)
            ## Conv 5
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)
            ## Conv 6
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)
            ## Conv 7
            x = BatchNormalization()(x)
            x = Activation("relu")(x)
            x = Conv2D(n_filter, (1, 1), padding="same")(x)
            ## Shortcut
            s  = Conv2D(n_filter, (1, 1), padding="same")(x_init)
            s = BatchNormalization()(s)

            ## Add
            x = Add()([x, s])
            return x

        def inter_block(x, n_filter, pool=True):
            x1 = conv_block(x, n_filter)
            c = x1

            ## Pooling
            if pool == True:
                x = MaxPooling2D((2, 2), (2, 2))(x1)

                return c, x
            else:
```

```
                x = AugumentedSubPixelShuffling(x1)
                return c

        n_filters = [16, 32, 64, 96, 128]
        inputs = Input((self.input_size, self.input_size, 1))

        c0 = inputs
        ## Encoder
        c1, p1 = inter_block(c0, n_filters[0])
        c2, p2 = inter_block(p1, n_filters[1])
        c3, p3 = inter_block(p2, n_filters[2])
        c4, p4 = inter_block(p3, n_filters[3])

        ## Bridge
        b1 = inter_block(p4, n_filters[4], pool=False)
        b2 = inter_block(b1, n_filters[4], pool=False)

        ## Decoder
        d1 = Conv2DTranspose(n_filters[3], (3, 3), padding="same",
strides=(2, 2))(b2)
        #d1 = UpSampling2D((2, 2))(b2)
        d1 = Concatenate()([d1, c4])
        d1 = inter_block(d1, n_filters[3], pool=False)

        d2 = Conv2DTranspose(n_filters[3], (3, 3), padding="same",
strides=(2, 2))(d1)
        #d2 = UpSampling2D((2, 2))(d1)
        d2 = Concatenate()([d2, c3])
        d2 = inter_block(d2, n_filters[2], pool=False)

        d3 = Conv2DTranspose(n_filters[3], (3, 3), padding="same",
strides=(2, 2))(d2)
        #d3 = UpSampling2D((2, 2))(d2)
        d3 = Concatenate()([d3, c2])
        d3 = inter_block(d3, n_filters[1], pool=False)

        d4 = Conv2DTranspose(n_filters[3], (3, 3), padding="same",
strides=(2, 2))(d3)
        #d4 = UpSampling2D((2, 2))(d3)
        d4 = Concatenate()([d4, c1])
        d4 = inter_block(d4, n_filters[0], pool=False)

        ## output
        outputs = Conv2D(self.n_classes, (1, 1),
padding="same")(d4)
        outputs = BatchNormalization()(outputs)
        outputs = Activation("softmax")(outputs)
```

```
        ## Model
        model = Model(inputs, outputs)
        return model


#-----------------------------------------------------------------
---------------
# DEEP GRADIENT ResNet Model

arch = Deep_Gradient_ResNet(input_size=128,n_classes=2)
model = arch.build_upsampling_block()
model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=metrics)

model_path = "Proposed_Model_t.h5"

checkpoint = ModelCheckpoint(model_path, verbose=1,
save_best_only=True)
callbacks = [checkpoint]


for idx in range(len(model.layers)):
  print(model.get_layer(index = idx).name)
import tensorflow
tensorflow.keras.utils.plot_model(model, show_shapes=True)


# # MODEL FITTING
history = model.fit(x_train, y_train_cat,
                    batch_size = 32,
                     verbose=1,
                     epochs=100,
                    validation_data=(x_test, y_test_cat)
                     )


#model.save("Proposed_model_t1.h5")
import keras
from matplotlib import pyplot as plt
plt.plot(history.history['recall'])
plt.plot(history.history['val_recall'])
plt.title('Model Recall')
plt.ylabel('Recall')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()


plt.plot(history.history['precision'])
plt.plot(history.history['val_precision'])
plt.title('Model Precision')
```

```
plt.ylabel('Precision')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()


from keras.models import load_model

model =
load_model("Proposed_model_t.h5",custom_objects={'dice_coef':dice_
coef},compile=False)


#----------------------------------------------------------------
---------------
"""MODEL PREDICTION"""

y_pred=model.predict(x_test)
y_pred_argmax=np.argmax(y_pred, axis=3)


#----------------------------------------------------------------
---------------
"""### PREDICTION ON TEST IMAGE"""


test_img_number =10
test_img = x_test[test_img_number]
ground_truth=y_test[test_img_number]
test_img_norm=test_img
test_img_input=np.expand_dims(test_img_norm, 0)
prediction = (model.predict(test_img_input))
predicted_img=np.argmax(prediction, axis=3)[0,:,:]

# PLOTTING
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title('Testing Image')
plt.imshow(test_img, cmap='gray')
plt.subplot(232)
plt.title('Testing Label')
plt.imshow(ground_truth[:,:,0], cmap='gray')
plt.subplot(233)
plt.title('Prediction on test image')
plt.imshow(predicted_img, cmap='gray')
plt.show()
```

**Classification**

```
import os
```

```
import cv2
import glob
import keras
import numpy as np
import random
np.random.seed(10)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
from imblearn.over_sampling import RandomOverSampler
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split


from google.colab.patches import cv2_imshow
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten,
Dropout
from keras.models import load_model
from keras.utils.image_dataset import image_dataset_from_directory
from sklearn.metrics import accuracy_score,classification_report,
confusion_matrix


SIZE = 64
# X_img = []
# Y_lab= []
# for directory_path in glob.glob("Type Of Crack/*"):
#     label = directory_path.split("\\")[-1]
#     # print(label)
#     for img_path in glob.glob(os.path.join(directory_path,
"*.jpg")):
#         print(img_path)
#         img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
#         img = cv2.resize(img, (SIZE, SIZE))
#         X = np.array(img)

#         X_img.append(X)
#         Y_lab.append(label)
# x = np.array(X_img)
# y = np.array(Y_lab)
x=np.load("x_arr.npy")
y=np.load("y_arr.npy")
x = x.reshape(-1, 64, 64, 1)


SIZE = 64
# X_img = []
# Y_lab= []
# for directory_path in glob.glob("Type Of Crack/*"):
```

```
#     label = directory_path.split("\\")[-1]
#     # print(label)
#     for img_path in glob.glob(os.path.join(directory_path,
"*.jpg")):
#         print(img_path)
#         img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
#         img = cv2.resize(img, (SIZE, SIZE))
#         X = np.array(img)

#         X_img.append(X)
#         Y_lab.append(label)
# x = np.array(X_img)
# y = np.array(Y_lab)
x=np.load("x_arr.npy")
y=np.load("y_arr.npy")
x = x.reshape(-1, 64, 64, 1)


print("x shape",x.shape)
print("y shape",y.shape)
# Normalize pixel values to between 0 and 1
x= x / 255.0
#Encode labels from text to integers.
le = preprocessing.LabelEncoder()
y = le.fit_transform(y)
list(le.classes_)
# one hot encode
Y_cat= to_categorical(y)
#Split data into test and train datasets
x_train, x_test, y_train, y_test = train_test_split(x, Y_cat,
test_size=0.1,random_state=42)
print('x_train_shape', x_train.shape)
print('x_test_shape', x_test.shape)
print('y_train_shape', y_train.shape)
print('y_test_shape', y_test.shape)
from keras.layers import Input, Add, Activation,
GlobalAveragePooling2D, GlobalMaxPooling2D,
BatchNormalization,multiply
from keras.models import Model

# Create the Residual Block
def residual_block(x, filters, kernel_size=(3, 3), strides=(1, 1),
activation='relu'):
    y = Conv2D(filters, kernel_size=kernel_size, strides=strides,
padding='same')(x)
    y = BatchNormalization()(y)
    y = Activation(activation)(y)
    y = Conv2D(filters, kernel_size=kernel_size, strides=strides,
padding='same')(y)
```

```
    y = BatchNormalization()(y)
    if strides != (1, 1) or x.shape[-1] != filters:
        x = Conv2D(filters, kernel_size=(1, 1), strides=strides,
padding='same')(x)
    return Add()([x, y])

# Define the number of classes
num_classes = 3

# Input layer
input_layer = Input(shape=(64, 64, 1))

# ResNet layers
res_block1 = residual_block(input_layer, filters=64)
res_block2 = residual_block(res_block1, filters=64)
res_block3 = residual_block(res_block2, filters=32)
res_block4 = residual_block(res_block3, filters=32)
res_block5 = residual_block(res_block4, filters=16)
res_block6 = residual_block(res_block5, filters=16)
res_block7 = residual_block(res_block6, filters=8)
res_block8 = residual_block(res_block7, filters=8)

# Attention mechanism
attention = GlobalAveragePooling2D()(res_block8)
attention = Dense(64, activation='relu')(attention)
attention = Dense(1, activation='sigmoid')(attention)
attention = multiply([res_block8, attention])

# Combine Global Average Pooling and Global Max Pooling
global_pooling = Add()([GlobalAveragePooling2D()(attention),
GlobalMaxPooling2D()(attention)])

# Flatten layer
flatten_layer = Flatten()(global_pooling)

# Dense layers
dense_layer = Dense(256, activation='relu')(flatten_layer)
output_layer = Dense(num_classes,
activation='softmax')(dense_layer)

# Create the model
model = Model(inputs=input_layer, outputs=output_layer)

# Compile the model with a categorical cross-entropy loss function
and an Adam optimizer
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()
```

```
import tensorflow
tensorflow.keras.utils.plot_model(model, show_shapes=True)
history =
model.fit(x_train,y_train,batch_size=32,epochs=150,validation_data
=(x_test,y_test))
# model.save("Model_TOC_84.h5")

model = load_model("Model_TOC_84.h5")

# Generate predictions on the test dataset
y_pred = model.predict(x_test)
# Convert the predicted probabilities to class labels
y_pred_classes = np.argmax(y_pred, axis=1)
y_tests=np.argmax(y_test, axis=1)
# np.save("y_preds_toc1.npy",y_pred_classes)
# np.save("y_test_toc1.npy",y_tests)
y_test= np.load("y_test_toc1.npy")
y_pred= np.load("y_preds_toc1.npy")

print("\n\tACCURACY SCORE\n\t*******************************\n")
print (f"\t{accuracy_score(y_tests,y_pred_classes)*100}")

# Print the classification report
print(classification_report(y_tests,y_pred_classes))

"Compute the confusion matrix"
cm = confusion_matrix(y_tests, y_pred_classes)
# Plot the confusion matrix
plt.imshow(cm, cmap='Oranges')
plt.colorbar()
plt.xlabel('predicted')
plt.ylabel('Original')
# Add the labels to the x and y axis
tick_marks = np.arange(3)
class_labels = ["Aligator", "Longitude", "Transverse"]
plt.title('Confusion Matrix')
plt.xticks(tick_marks, class_labels)
plt.yticks(tick_marks, class_labels)
# Add the values to the confusion matrix
for i in range(3):
  for j in range(3):
    plt.text(j, i, cm[i, j], ha='center', va='center',
color='black')
#plt.imshow()
plt.show()
def convert_to_meters_alli(height_pixels, width_pixels,
```

```
pixel_size):
    height_m = height_pixels * pixel_size
    width_m = width_pixels * pixel_size
    return height_m, width_m

def convert_to_meters(Lenght_pixels, pixel_size):
    Lenght_m = Lenght_pixels * pixel_size
    return Lenght_m

def area(height_pixels, width_pixels):
    area1=height_m*width_m
    return area1

def blendImages(img1, img2):
    blended_image = cv2.addWeighted(img1, 0.7, img2, 0.3, 0)
    return blended_image
```

**Prediction and Sizing**

```
# ----------------------------------------
# Alligator
# ----------------------------------------

image_path_ori =
"crack500/original/CRACK500_20160222_164141_1_721.jpg"
image_path_seg =
"crack500/mask/CRACK500_20160222_164141_1_721.jpg"


image_1 = cv2.imread(image_path_seg,0)
img_1 = cv2.resize(image_1, (SIZE, SIZE))

img_1[img_1 > 0] = 255
X_1 = np.array(img_1)

# normalize image to range between 0 and 1
X_new=X_1 / 255.0
img_full_1= X_new.reshape((X_new.shape[0], X_new.shape[1], 1))

# expand the dimensions of image to match model input shape
img_full_res = np.expand_dims(img_full_1, axis=0)

# pass preprocessed image to model for prediction
model_outputs = model.predict(img_full_res)

# Checking the absolute predicted probabilities to class labels
ary=model_outputs
max_value_1 = np.amax(ary)
max_index_1 = np.argmax(ary)
```

```
string_array_1 = np.array(["Aligator", "Longitude", "Transverse"])
float_array_2 = ary
max_index_1 = np.argmax(float_array_2)
result_val_1 =
str(f"{string_array_1[max_index_1]}:{str(max_value_1)}")
print(result_val_1)

if max_index_1 == 0:
  # read image
  image = cv2.imread(image_path_seg)
  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]
  # get contours
  result = image.copy()
  result1 = image.copy()
  img_1 = cv2.resize(result, (SIZE, SIZE))
  img_1[img_1 > 0] = 140
  # Convert the segmented image to binary format
  binary_image = np.where(img_1 == 140, 1, 0)
  img_2 = cv2.resize(result1, (SIZE, SIZE))
  img_2[img_2 > 0] = 140
  # Convert the segmented image to binary format
  binary_image1 = np.where(img_2 == 140, 1, 0)
  # Calculate the area of the crack
  crack_area_pixels = np.sum(binary_image)
  # crack_area_meters = crack_area_pixels * 0.0001
  crack_area_meters = crack_area_pixels * 0.001
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]
  for cntr in contours:
      x, y, w, h = cv2.boundingRect(cntr)
      # Convert height and width to meters using the conversion
function
      height_m, width_m = convert_to_meters_alli(h, w,
pixel_size=0.02)
      area_val_ali=  area(height_m, width_m)
      # Display the height and width in meters on the image
      text11 = f"AREA : {crack_area_meters:.1f} Sq m"
      text12 = f"Pixel : {crack_area_pixels:.1f} px"
  cv2.putText(result, text11, (45,20), cv2.FONT_HERSHEY_SIMPLEX,
 0.5, (125, 246, 55),  2)
  cv2.putText(result1, text12, (45,20), cv2.FONT_HERSHEY_SIMPLEX,
 0.5, (125, 246, 55),  2)


elif max_index_1 == 1:
```

```
   # read image
  image = cv2.imread(image_path_seg)
  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]
  # get contours
  result = image.copy()
  result1 = image.copy()
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]
  for cntr in contours:
      x, y, w, h = cv2.boundingRect(cntr)
      # Convert height and width to meters using the conversion
function
      Lenght_m = convert_to_meters(h,pixel_size=0.005)
      # Display the height and width in meters on the image
      text = f"Length: {Lenght_m:.2f}m"
      cv2.putText(result, text, (80,15), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 255,0 ), 2)
      text12 = f"Pixel: {h:.2f}px"
      cv2.putText(result1, text12, (80,15),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,0 ), 2)


else:
  # read image
  image = cv2.imread(image_path_seg)
  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]
  # get contours
  result = image.copy()
  result1 = image.copy()
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]
  for cntr in contours:
      x, y, w, h = cv2.boundingRect(cntr)
      # Convert height and width to meters using the conversion
function
      Lenght_m = convert_to_meters(w,pixel_size=0.005)
      # Display the height and width in meters on the image
      text = f"Length: {Lenght_m:.2f}m"
      cv2.putText(result, text, (x, y-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,0 ), 2)
      text12 = f"Pixel: {w:}px"
```

```
      cv2.putText(result1, text12, (x, y-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,0 ), 2)

# save resulting image
cv2.imwrite('images_up.jpg', result)
cv2.imwrite('images_up1.jpg', result1)
# Blend the images
img1 = cv2.imread("images_up.jpg")
img1_1 = cv2.imread("images_up1.jpg")
img2 = cv2.imread(image_path_ori)
newImage = blendImages(img1, img2)
newImage_px = blendImages(img1_1, img2)
# PLOTTING
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title(result_val_1)
plt.imshow(cv2.cvtColor(newImage_px, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(232)
plt.title(result_val_1)
plt.imshow(cv2.cvtColor(newImage, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()


# ----------------------------------------
# Longitude
# ----------------------------------------
image_path_ori ="Video
Frame/Longitude/Original/02977_jpg.rf.471921b4dd22ee6a6b3a7e61fd25
452e.jpg"
image_path_seg ="Video
Frame/Longitude/Segment/02977_jpg.rf.471921b4dd22ee6a6b3a7e61fd254
52e.jpg"


image_1 = cv2.imread(image_path_seg,0)
img_1 = cv2.resize(image_1, (SIZE, SIZE))

img_1[img_1 > 0] = 255
X_1 = np.array(img_1)

# normalize image to range between 0 and 1
X_new=X_1 / 255.0

img_full_1= X_new.reshape((X_new.shape[0], X_new.shape[1], 1))
# print(img_full_1.shape)

# expand the dimensions of image to match model input shape
img_full_res = np.expand_dims(img_full_1, axis=0)
```

```
# print(img_full_res.shape)

# pass preprocessed image to model for prediction
model_outputs = model.predict(img_full_res)

# Checking the absolute predicted probabilities to class labels
ary=model_outputs
max_value_1 = np.amax(ary)
max_index_1 = np.argmax(ary)
string_array_1 = np.array(["Aligator", "Longitude", "Transverse"])
float_array_2 = ary
max_index_1 = np.argmax(float_array_2)
result_val_1 =
str(f"{string_array_1[max_index_1]}:{str(max_value_1)}")


if max_index_1 == 0:
  # read image
  image = cv2.imread(image_path_seg)
  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]
  # get contours
  result = image.copy()
  result1 = image.copy()
  img_1 = cv2.resize(result, (SIZE, SIZE))
  img_1[img_1 > 0] = 140
  # Convert the segmented image to binary format
  binary_image = np.where(img_1 == 140, 1, 0)
  img_2 = cv2.resize(result1, (SIZE, SIZE))
  img_2[img_2 > 0] = 140
  # Convert the segmented image to binary format
  binary_image1 = np.where(img_2 == 140, 1, 0)
  # Calculate the area of the crack
  crack_area_pixels = np.sum(binary_image)
  crack_area_pixels1 = np.sum(binary_image1)
  # crack_area_meters = crack_area_pixels * 0.0001
  crack_area_meters = crack_area_pixels * 0.001
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]
  for cntr in contours:
      x, y, w, h = cv2.boundingRect(cntr)
      # Convert height and width to meters using the conversion
function
      height_m, width_m = convert_to_meters_alli(h, w,
pixel_size=0.02) # assuming pixel size is 0.2 mm
      area_val_ali=  area(height_m, width_m)
```

```
      # Display the height and width in meters on the image
      text11 = f"AREA : {crack_area_meters:.1f} Sq m"
      # Display the height and width in meters on the image
      text12 = f"Pixel : {crack_area_pixels1:} px"
  # cv2.putText(result, text11, (30,7), cv2.FONT_HERSHEY_SIMPLEX,
 0.5, (125, 246, 55),  1)
  cv2.putText(result, text11, (30,7), cv2.FONT_HERSHEY_SIMPLEX,
 0.25, (125, 246, 55),  1)
  cv2.putText(result1, text12, (30,7), cv2.FONT_HERSHEY_SIMPLEX,
 0.25, (125, 246, 55),  1)
elif max_index_1 == 1:
   # read image
  image = cv2.imread(image_path_seg)

  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]

  # get contours
  result = image.copy()
  result1 = image.copy()
  # Assuming you have already obtained the contours using
cv2.findContours()
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]

  # Minimum length threshold
  min_length = 100  # Adjust this value according to your needs

  # Filter out small contours
  filtered_contours = []
  for cntr in contours:
    if cv2.arcLength(cntr, closed=True) >= min_length:
        filtered_contours.append(cntr)
        x, y, w, h = cv2.boundingRect(cntr)
        # Convert height and width to meters using the conversion
function
        Lenght_m = convert_to_meters(h,pixel_size=0.02) # assuming
pixel size is 0.2 mm
        # Lenght_m = convert_to_meters(h,pixel_size=0.005) #
assuming pixel size is 0.2 mm
        # Display the height and width in meters on the image
        text = f"Length: {Lenght_m:.2f}m"
        text12 = f"Pixel: {crack_area_pixels:}px"
        cv2.putText(result, text, (x, y+5),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 0.25, (0, 255,0 ), 1)
        cv2.putText(result1, text12, (x, y+5),
cv2.FONT_HERSHEY_SIMPLEX, 0.25, (0, 255,0 ), 1)
else:
  # read image
  image = cv2.imread(image_path_seg)

  # convert to grayscale
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

  # threshold
  thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY)[1]

  # get contours
  result = image.copy()
  result1 = image.copy()
  # Assuming you have already obtained the contours using
cv2.findContours()
  contours = cv2.findContours(thresh, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  contours = contours[0] if len(contours) == 2 else contours[1]

  # Minimum length threshold
  min_length = 100  # Adjust this value according to your needs

  # Filter out small contours
  filtered_contours = []
  for cntr in contours:
    if cv2.arcLength(cntr, closed=True) >= min_length:
        filtered_contours.append(cntr)
        x, y, w, h = cv2.boundingRect(cntr)
        # Convert height and width to meters using the conversion
function
        Lenght_m = convert_to_meters(w,pixel_size=0.02) # assuming
pixel size is 0.2 mm
        # Lenght_m = convert_to_meters(h,pixel_size=0.005) #
assuming pixel size is 0.2 mm
        # Display the height and width in meters on the image
        text = f"Length: {Lenght_m:.2f}m"
        text12 = f"Pixel: {w:.2f}px"
        cv2.putText(result, text, (x, y+5),
cv2.FONT_HERSHEY_SIMPLEX, 0.25, (0, 255,0 ), 1)
        cv2.putText(result1, text12, (x, y+5),
cv2.FONT_HERSHEY_SIMPLEX, 0.25, (0, 255,0 ), 1)


# save resulting image
cv2.imwrite('images_up.jpg', result)
cv2.imwrite('images_up1.jpg', result1)
```

```
# Load the first image
img1 = cv2.imread("images_up.jpg")
imgs1 = cv2.resize(img1, (448, 448),
interpolation=cv2.INTER_LINEAR)
# Load the second image
img3 = cv2.imread(image_path_ori)
img2 = cv2.resize(img3, (448, 448),
interpolation=cv2.INTER_LINEAR)
# Blend the images
newImage = blendImages(imgs1, img2)


# Load the first image
img1 = cv2.imread("images_up1.jpg")
imgs1 = cv2.resize(img1, (448, 448),
interpolation=cv2.INTER_LINEAR)
# Load the second image
img3 = cv2.imread(image_path_ori)
img2 = cv2.resize(img3, (448, 448),
interpolation=cv2.INTER_LINEAR)
# Blend the images
newImage_px = blendImages(imgs1, img2)


# PLOTTING
plt.figure(figsize=(12, 8))
plt.subplot(231)
plt.title(result_val_1)
plt.imshow(cv2.cvtColor(newImage_px, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.subplot(232)
plt.title(result_val_1)
plt.imshow(cv2.cvtColor(newImage, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```