# EVALUATING THE PERFORMANCE OF CNN-BASED ALGORITHMS FOR BUILDING EXTRACTION IN COMPARISON WITH CLASSICAL IMAGE PROCESSING METHODS

# CNN TABANLI ALGORİTMALARIN KLASİK GÖRÜNTÜ İŞLEME YÖNTEMLERİ İLE KIYASLANARAK BİNA ÇIKARIMI PERFORMANSININ DEĞERLENDİRİLMESİ

**ZEINAB BAYAT**

**Assoc. Prof. Dr BURCU GÜLDÜR ERKAL**

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Sience in Civil Engineering.

2024

## ABSTRACT

**EVALUATING THE PERFORMANCE OF CNN-BASED ALGORITHMS FOR BUILDING EXTRACTION IN COMPARISON WITH CLASSICAL IMAGE PROCESSING METHODS**

**Zeinab Bayat**

**Master of Science, Department of Civil Engineering**

**Supervisor: Assoc. Prof. BURCU GÜLDÜR ERKAL**

**June 2024, 95 pages**

In the early morning of Local time at 04:17 AM, on the sixth of February 2023, (01:17 UTC), a powerful earthquake measuring Mw 7.8 struck Syria's northern and western regions as well as the southern and central portions of Turkey.. The serious problem of closed roads laden with debris, arising after the devastating earthquake in Turkey, significantly hampers disaster response and recovery efforts. This thesis emphasizes the growth and evaluation of algorithms for the detection and classification of closed roads due to debris generated after the earthquake, using images captured by drones. The study employs classic image processing methods implemented in Python along with Convolutional neural network (CNN) based algorithms. The unique dataset containing images captured after the earthquake provides a detailed picture of extensive road blockages resulting from the disaster's aftermath.

An important contribution of this research the emergence of an intuitive and A GUI (Graphical User Interface) that is intuitive and easy for users to navigate. This GUI panel serves as a comprehensive platform showcasing a wide range of classical and deep learning image processing methods, allowing users to interactively visualize the results of these methods. This interactive tool not only simplifies access to algorithmic outputs but also empowers disaster response teams and decision-makers to make data-driven and effective decisions.

The principal aim of of This research is to to assess how accurate and effective the proposed algorithms are in identifying the nature and prevalence of road blockages. By distinguishing between road segments filled with debris and those left open, this study provides critical information to expedite disaster relief logistics and prioritize recovery operations.

The findings presented in this thesis contribute to enhancing the disaster management toolkit by expediting data-driven decision processes for post-earthquake recovery operations, thereby contributing to the overall resilience of affected communities.

**Keywords**: Road Blockage Detection, Earthquake Impact, Disaster Response, Convolutional neural networks, CNN Algorithms, Image Processing, Python Programming, Disaster Relief, Post-Earthquake Assessment, Seismic Event, Turkish Earthquake, Aerial Observation, Graphical User Interface (GUI), Real-Time Results Visualization, Image Processing Methods, Decision Support Interface, Humanitarian Aid, Machine Learning Algorithms, Computer Vision, Human Intervention, Crisis Management, Geographic Information System-based Decision Support.

# ÖZET

**CNN TABANLI ALGORİTMALARIN KLASİK GÖRÜNTÜ İŞLEME YÖNTEMLERİ İLE KIYASLANARAK BİNA ÇIKARIMI PERFORMANSININ DEĞERLENDİRİ MESİ Zinab Bayat**

**İnşaat Mühendisliği Bölümü**

**Tez Danışmanı: Assoc. Prof. BURCU GÜLDÜR ERKAL**

**Haziran 2024, 95 sayfa**

6 Şubat 2023 tarihinde, yerel saatle sabah 04:17'de (01:17 UTC), Mw 7.8 büyüklüğünde güçlü bir deprem Suriye'nin kuzey ve batı bölgeleri ile Türkiye'nin güney ve orta kesimlerini vurdu. Türkiye'de meydana gelen bu yıkıcı depremin ardından enkazla dolu kapalı yollar, afet müdahale ve iyileştirme çabalarını ciddi şekilde engellemektedir. Bu tez, dronlarla çekilen görüntüler kullanılarak deprem sonrası oluşan enkaz nedeniyle kapalı yolların tespiti ve sınıflandırılması için algoritmaların geliştirilmesi ve değerlendirilmesine odaklanmaktadır. Çalışmada, Python'da uygulanan klasik görüntü işleme yöntemleri ile Konvolüsyonel sinir ağı (CNN) tabanlı algoritmalar kullanılmaktadır. Deprem sonrası çekilen görüntüleri içeren benzersiz veri seti, felaketin ardından oluşan kapsamlı yol tıkanıklıklarının detaylı bir resmini sunmaktadır.

Çalışma, Python dilinde uygulanan klasik görüntü işleme yöntemleri ile birlikte Convolutional neural network (CNN) tabanlı algoritmaları kullanmaktadır. Depremin ardından yakalanan görüntüler içeren benzersiz bir veri kümesi, felaketin yan etkileri olarak oluşan geniş yol tıkanıklıklarının ayrıntılı bir resmini sunmaktadır.

Bu araştırmanın önemli bir katkısı, sezgisel ve kullanıcı dostu bir Grafiksel Kullanıcı Arayüzü (GUI) paneli oluşturulmasıdır. Bu GUI paneli, klasik ve derin öğrenme görüntü işleme yöntemlerinin geniş bir yelpazesini sergileyen kapsamlı bir platform olarak hizmet vermektedir ve kullanıcılara bu yöntemlerin sonuçlarını etkileşimli olarak görüntüleme imkanı sunmaktadır. Bu etkileşimli araç, yalnızca algoritmik çıktılara erişimi basitleştirmekle kalmayıp aynı zamanda felaket müdahale ekiplerini ve karar vericileri veri odaklı ve etkili kararlar almada yeteneklendirmektedir.

Bu araştırmanın temel amacı, önerilen algoritmaların yol tıkanıklarının doğasını ve yaygınlığını tanımlamada ne kadar doğru ve etkili olduğunu değerlendirmektir. Enkazla dolu yol segmentleri ile açık kalanlar arasındaki ayrımı yaparak, bu çalışma felaket yardım lojistiğini hızlandırmak ve iyileştirme operasyonlarını önceliklendirmek için kritik bilgiler sunmaktadır.

Bu tezde sunulan bulgular, deprem sonrası iyileştirme operasyonlarının veri odaklı karar süreçlerini hızlandırarak etkilenen toplulukların genel direncine katkıda bulunarak felaket yönetimi araç setini geliştirmektedir.

**Anahtar Kelimeler**: Yol Tıkanıklık Tespiti, Deprem Etkisi, Felaket Müdahalesi, Evrişimli Sinir Ağları, CNN Algoritmaları, Görüntü İşleme, Python Programlama, Felaket Yardımı, Deprem Sonrası Değerlendirme, Sismik Olay, Türkiye Depremi, Hava Gözlemi, Grafiksel Kullanıcı Arayüzü (GUI), Gerçek Zamanlı Sonuçlar Görüntüleme, Görüntü İşleme Yöntemleri, Karar Destek Arayüzü, İnsani Yardım, Makine Öğrenme Algoritmaları, Bilgisayarlı Görü, İnsani Müdahale, Kriz Yönetimi, Coğrafi Bilgi Sistemine Dayalı Karar Desteği.

# ACKNOWLEDGEMENTS

The MSC journey has been one of the most extraordinary experiences of my life, marking the first chapter of my scientific career. It’s not the end but the beginning. Through this journey, I have built a strong foundation for the next phase, learning the importance of patience, adaptability, idea management, and effective communication.

First of all, I would want to express my profound thanks to my supervisor Assoc. Prof. BURCU GÜLDÜR ERKAL. Her consistent support, guidance, and valuable supervision during my MSC degree were invaluable. Thank you for helping me navigates and completes this project.

I would also like to express my gratitude to Prof. Dr. Baki Öztürk, Assoc. Prof. Dr. Alper Aldemir, Assoc. Prof. Dr. M. Abdullah Sandıkkaya, and Doc. Dr. Zeynep Değer for providing me the opportunity to defend my master’s thesis.

Special thanks go to my dear friend and life companion, Ahmad Salmanoghli Khiavi. Your support and assistance were indispensable. I couldn't have done this without you. Your help was crucial in getting through this long journey.

Finally, my heartfelt thanks to my parents. They set me off on this challenging yet rewarding path a long time ago. Their encouragement and support have been the backbone of my journey.

This MSC journey is just the start of my scientific story, a journey where I learned to be patient, adapt, manage ideas, and express thoughts effectively. I look forward to the next chapters with confidence and excitement.

# CONTENTS

# TABLES

# FIGURES

# SYMBOLS AND ABBREVIATIONS

**Abbreviations**

| | |
|---|---|
| DIP | Digital image processing |
| DL | Deep Learning |
| CNN | Convolutional Neural Network |
| IP | Image Processing |
| AI | Artificial Intelligence |
| ML | Machine Learning |
| ANNs | Artificial Neural Networks |
| RNNs | Recurrent Neural Networks |
| GANs | Generative Adversarial Networks |
| SLTMs | Long Short-Term Memory Networks |
| TL | Transfer Learning Models |
| CapsNets | Capsule Networks |
| RESNET | Residual Network |
| SENET | Squeeze-and-Excitation Network |
| YOLO | You Only Look Once |
| PIL | Python Imaging Library |
| SLIC | Simple Linear Iterative Clustering |
| ACWE | Active contours without edges |
| QGIS | Quantum Geographic Information System |
| COCO | Common Objects in Context |
| FPN | Feature Pyramid Network |
| PAN | Path Aggregation Network |
| TFRecord | TensorFlow Record |
| Kitti | Karlsruhe Institute of Technology and Toyota Technological Institute |
| Img | image |

| | |
|---|---|
| MMDetection | ultimedia Detection |
| GPUs | Graphics Processing Units |
| Soft-NMS | Soft Non-Maximum Suppression |
| mAPval | Average Precision validation score |
| FLOPs | Floating Point Operations Per Second |
| ONNX | Open Neural Network Exchange |
| CPU | Central Processing Unit |
| Ms | milliseconds |
| params(M) | number of parameters in millions |
| RPN | region proposal network |
| Seg | segmentation |
| Val | Validation |
| Cls | class |
| IoU | Intersection over Union |
| lr/pg | learning rates/ parameter groups |
| obj | object |
| dfl | Degree of Freedom Loss |
| GUI | Graphical User Interfac |

# 1. INTRODUCTION

## 1.1. Image processing

Digital image processing encompasses the manipulation of digital images utilizing a digital computer. Image processing encompasses the analysis and enhancement of visual information for improved interpretation [16]. Specifically, digital image processing (DIP) involves the Systematic examination and analysis of images using computer systems. On the other hand, computer vision is focused on developing models, extracting data, and deriving data from images [13].

A digital image, comprising finite elements and values, undergoes this processing, but authors differ on its boundaries with image analysis and computer vision. Some define it as a discipline where images serve as both input and output. For instance, even computing the average intensity, yielding a single number, might not be considered image processing. On the other hand, computer vision aims to replicate human vision in computers, enabling them to learn, draw inferences, and initiate actions based on visual stimuli [16].

Classical image processing has varying applications; in this study, the segmentation technique was employed to isolate cracks based on the gray-level differences between the cracks and their background. Additionally, the Canny iterative method was utilized to detect crack edges, leveraging the linear characteristics of the cracks. The Otsu method, along with multiple filtering techniques, was also applied to identify cracks in concrete structures. While these algorithms produced satisfactory experimental results, they were tested on images with simple backgrounds, without considering obstacles. Therefore, these methods might not be suitable for detecting debris in more complex backgrounds. Recently, many researchers have shifted towards using deep learning (DL) instead of traditional image processing algorithms, highlighting the trend towards advanced computer vision and deep learning techniques in the field Contemporary researchers are embracing advanced technologies, particularly computer vision and deep learning, like DL-based image processing, to enhance debris detection capabilities in complex scenarios with intricate backgrounds and potential obstacles [2].

Computer vision has transformed into a comprehensive field encompassing tasks ranging from the acquisition of raw data to the extraction of visual patterns and the interpretation of data [13]. The spectrum from image processing to computer vision encompasses various levels of complexity. At the foundational level, basic operations such as image preprocessing are performed. Moving up the hierarchy, tasks involve segmentation, description, and classification of objects within images. Higher-level processing delves into interpreting recognized objects, similar to image analysis, and, at the extreme, performing cognitive functions associated with vision. The field lacks clear-cut boundaries, prompting a paradigm that considers low, mid, and high-level computerized processes [16].

### 1.1.1. The image processing pipeline

The subsequent stages outline the fundamental processes within the image processing pipeline

*a) Acquisition and Storage:*

Capture the image using a device and store it on a device (e.g., hard disk) in a specific file format (e.g., JPEG).

*b) Load into Memory and Save to Disk:*

Read the image from the disk into memory, store it using a data structure (e.g., numpy array), and possibly serialize the data structure into an image file after applying algorithms.

*c) Manipulation, Enhancement, and Restoration:*

Execute preprocessing algorithms for tasks like image transformation, quality enhancement (e.g., de-blurring), and restoration from noise degradation.

*d) Segmentation:*

Divide the image to extract objects of interest, facilitating the isolation and identification of specific components.

*e) Information Extraction/Representation:*

Represent the image in an alternative form, involving hand-crafted feature descriptors (e.g., HOG descriptors) or automatic feature learning (e.g., deep learning).

*f) Image Understanding/Interpretation:*

Utilize the alternative representation to comprehend the image, performing tasks like image classification (e.g., identifying human presence) and object recognition (e.g., locating objects with bounding boxes) [17].

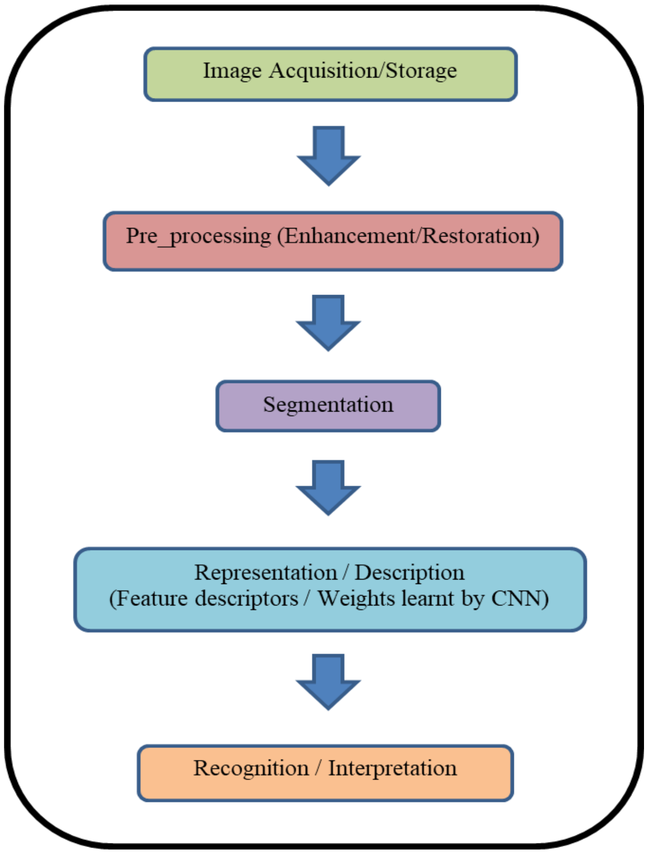


Image Acquisition/Storage
Pre_processing (Enhancement/Restoration)
Segmentation
Representation / Description
(Feature descriptors / Weights learnt by CNN)
Recognition / Interpretation

**Figure 1.1** Image processing pipeline

### 1.1.2. Civil Engineering Approach in Image Processing

The advancement in computer technology has significantly impacted diverse scientific and technological domains, enabling researchers to expand these techniques across various domains. Despite this progress, the construction industry falls behind in utilizing digitization to offer reliable data and improve decision-making processes, trailing other industries in this aspect. Artificial Intelligence and image processing (IP) are significantly influencing the construction sector lately. Both technologies possess distinct applications and potential within various facets of the construction industry. The ensuing sections will delineate the advancements in each of these technologies and elucidate their respective applications in the construction domain [1].

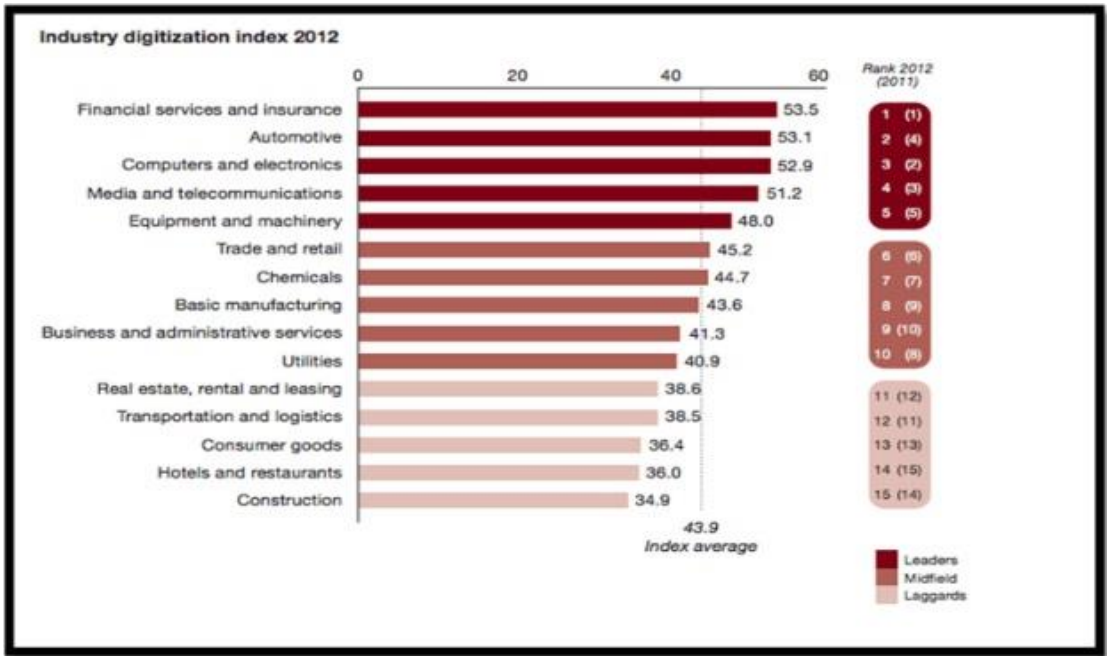


**Figure 1.2**. Industry digitization index [1]

The civil industry stands out as the least digitized sector, facing challenges such as climatic, environmental, economic, and sociopolitical complexities in civil projects, making digitization challenging. Over the past decade, civil engineers have increasingly focused on Simulations using computers and tracking systems that utilize image processing, artificial intelligence (AI), machine learning (ML), and other advanced technologies. These innovations are considered cost-effective and non-intrusive methods for collecting and analyzing image and video data in numerous construction firms [1-12].

The utilization of 3D imaging technologies in construction has led to various targeted applications by researchers. These applications include construction management, progress monitoring [10, 11], safety [18], quality control [18, 19] and building extraction [14].

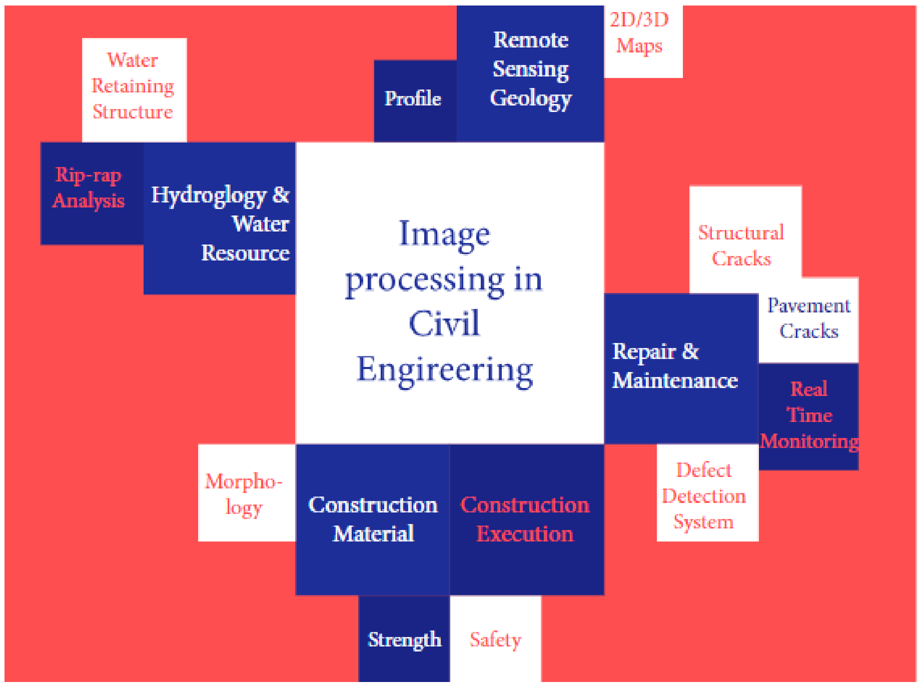


**Figure 1.3** Image processing applications in civil engineering [15]

## 1.2. Deep learning

In history, inventors, inspired by ancient Greek myths, envisioned creating thinking machines embodied by figures like Pygmalion and Daedalus. The advent of programmable computers in 1842, as speculated by Lovelace, fueled discussions about machines acquiring intelligence. Today, Artificial Intelligence (AI) is a dynamic field, addressing diverse practical needs. AI automates tasks, interprets speech/images, aids medical diagnoses, and supports scientific inquiries. In its early stages, AI excelled at computationally straightforward challenges but faced a true test in solving tasks performed effortlessly by humans, requiring intuitive navigation, such as recognizing spoken words or facial features in images.

Representation learning involves determining an optimal visualization of the incoming data, a key challenge in deep learning. The goal is to create efficient representations that capture essential features, facilitating tasks like image recognition and language understanding. Deep learning addresses this by constructing complex concepts from simpler ones. For instance, an image of a person is represented by combining basic elements like corners and contours, defined in terms of edges. This process enables the system to handle intricate variations in data, enhancing its ability to extract meaningful information [20].

### 1.2.1. What is deep learning?

Machine Learning (ML) pursues generalization through training on data, particularly vital for tasks like image classification. Traditional ML encounters challenges such as computational costs, limited learning progress, and dependence on human expertise. In contrast, Deep learning models show promise in overcoming these limitations and improving generalization in intricate tasks.

A subtype of machine learning called deep learning uses artificial neural networks (ANNs) of many layers inspired by the brain's structure. Each layer uses the output from the previous one, utilizing ANNs for feature extraction, pattern recognition, and abstraction development. Both supervised and unsupervised deep learning techniques are possible (e.g., pattern analysis) and employs gradient-descent algorithms to learn hierarchical representations. By depicting the universe as a layered hierarchy of notions, it acquires strength, where every idea relates to less complicated ones, forming a flexible abstraction hierarchy. In the context of image classification architecture, showcasing its adaptability and efficiency in processing complex information problem, a Deep learning model incrementally learns image classes through its hidden layer.

Initially, the system automatically discerns basic attributes like light or dark areas, progressing to more advanced features such as edges. Subsequently, it identifies the most intricate features, like shapes, facilitating classification. Each node or neuron encapsulates a specific facet of the overall image, collectively portraying the entire image comprehensively. The nodes effectively capture and represent the full complexity of the image. Furthermore, every node and neuron in the network is assigned weights, signifying their significance in influencing the output's strength. These weights are subject to adjustment during the model development phase, allowing for refinement and optimization of the neural network [17].

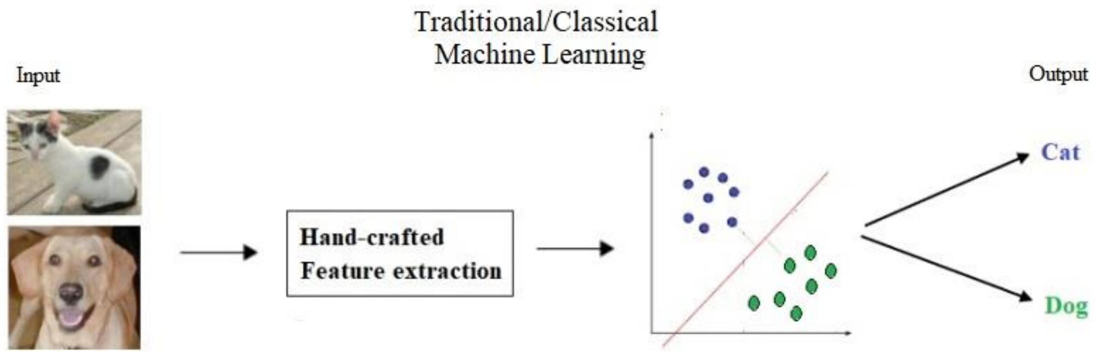

Traditional/Classical Machine Learning
Input
Hand-crafted Feature extraction
Output
Cat
Dog

**Figure 1.4** Traditional ML feature extraction [17]

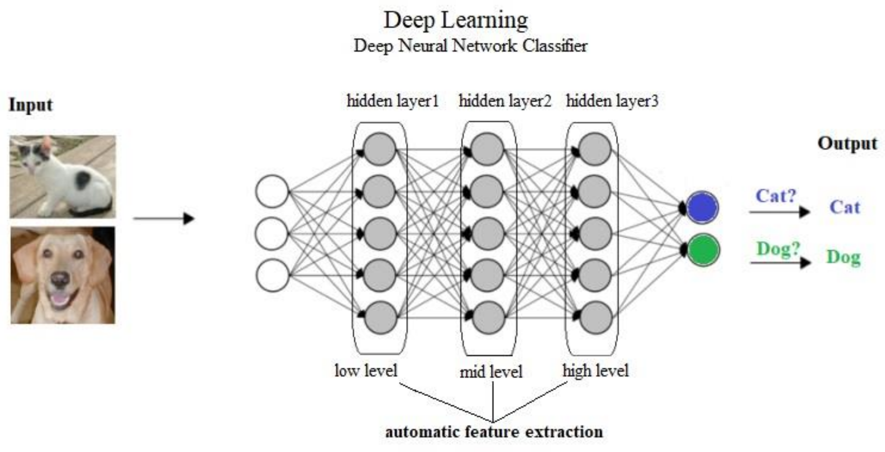


**Figure 1.5** Deep learning feature extraction [17]

### 1.2.2 Subsets of Deep Learning in Focus

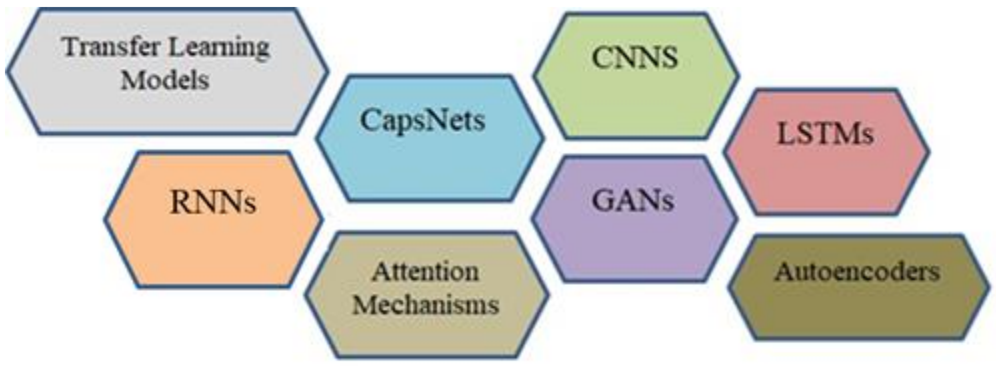


**Figure 1.6**. Deep learning subsets

a) Convolutional neural networks (CNNs):

   1. Focus on spatial hierarchies, primarily used in image-related tasks.
   2. Key architecture for image classification, object detection, and facial recognition [20].

b) Recurrent neural networks (RNNs):

   1. Specialized for consecutive data processing, such as time-series and natural language.

2. Suitable for tasks involving temporal dependencies [22].

c) Generative Adversarial Networks (GANs):

1. GANs comprise a generator and discriminator involved in an adversarial framework. The generator produces realistic data, while the discriminator identifies and differentiates real samples from generated ones.

2. GANs are utilized across a wide array of domains, spanning from image synthesis and style transfer to data augmentation. [23].

d) Long Short-Term Memory Networks (LSTMs):

1. These are a kind of recurrent neural network (RNN) created to solve the issue of the vanishing gradient in conventional RNNs [24].

e) Auto encoders:

1. Auto encoders, neural networks designed for unsupervised learning, comprise an encoder tasked with input data compression into a latent representation, followed by a decoder that reconstructs the original data. Widely applied, they excel in tasks like data de noising, anomaly detection, and feature learning, proving valuable across diverse domains [25].

f) Transfer Learning Models(TL):

1. Transfer Learning (TL) is recognized for its ability to establish connections between supplementary testing and training samples, leading to quicker outputs with effective results.

2. TL is specifically suitable when there is a scarcity of training data for the target task [26].

g) Attention Mechanisms:

1. They facilitate focusing on relevant parts of input data, enhancing model performance. For an in-depth understanding of attention mechanisms.

2. Play a crucial role in Deep learning, acting as a resource allocation scheme to address information flow challenges [27].

h) Capsule Networks (Caps Nets):

1. They are a neural network architecture type intended to solve constraints in traditional convolutional neural networks (CNNs).

2. The key innovation of Capsule Networks lies in the use of capsules, which are groups of neurons representing specific features and spatial relationships. Unlike neurons in traditional networks, capsules consider the pose or orientation of features, providing a more robust representation [28].

#### 1.2.2.1. Exploring Convolutional Neural Networks (CNNs)

In the realm of computer vision and Deep learning, a diverse set of convolutional neural network (CNN) algorithms assumes pivotal roles in various applications. Highlighted in the figure below are these algorithms with distinct architectural nuances, contributing significantly to advancements in image processing, object detection, and pattern recognition [30,31,46].

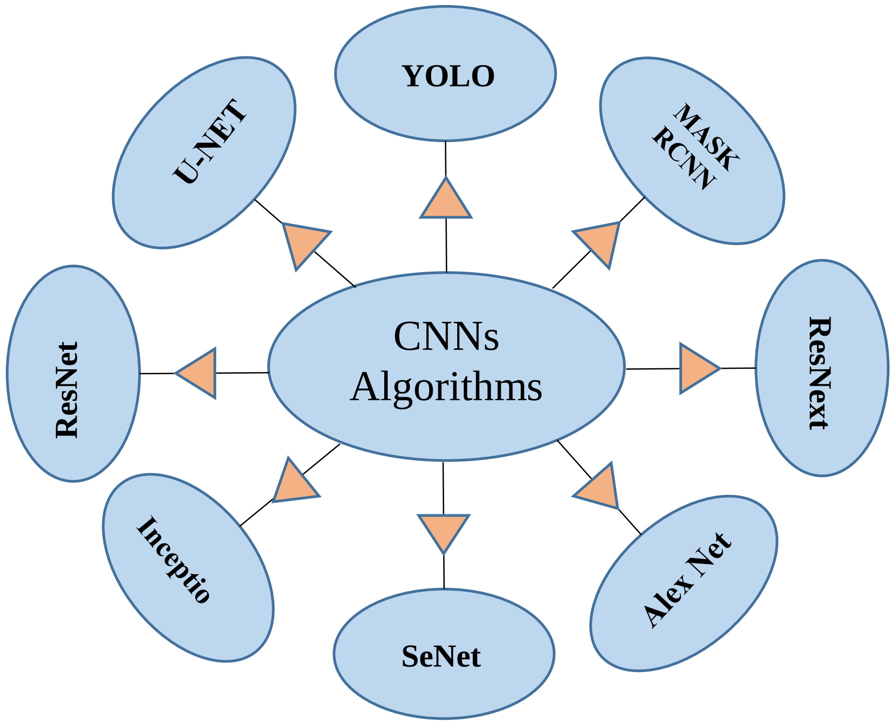


**Figure 1.7** CNNs algorithms

#### 1.2.2.2. An Introductory Guide to CNN Architecture

The CNN architecture comprises multiple layers, with a key component being the convolutional layer. This layer uses convolutional filters, or kernels, to convolve the input image, generating an output feature map. Kernels have discrete values called kernel weights, initialized randomly during training and adjusted to extract significant features. The convolutional operation involves sliding the kernel over the image, calculating dot products, and creating a feature map. Padding and stride impact the output size, with padding crucial for preserving border information. Convolutional layers offer benefits like sparse connectivity, reducing required weights and memory, and weight sharing, streamlining training time by becoming familiar with a particular set of weights for the entire input matrix. This architectural design enhances efficiency in memory use and computational costs compared to fully connect neural networks [30].

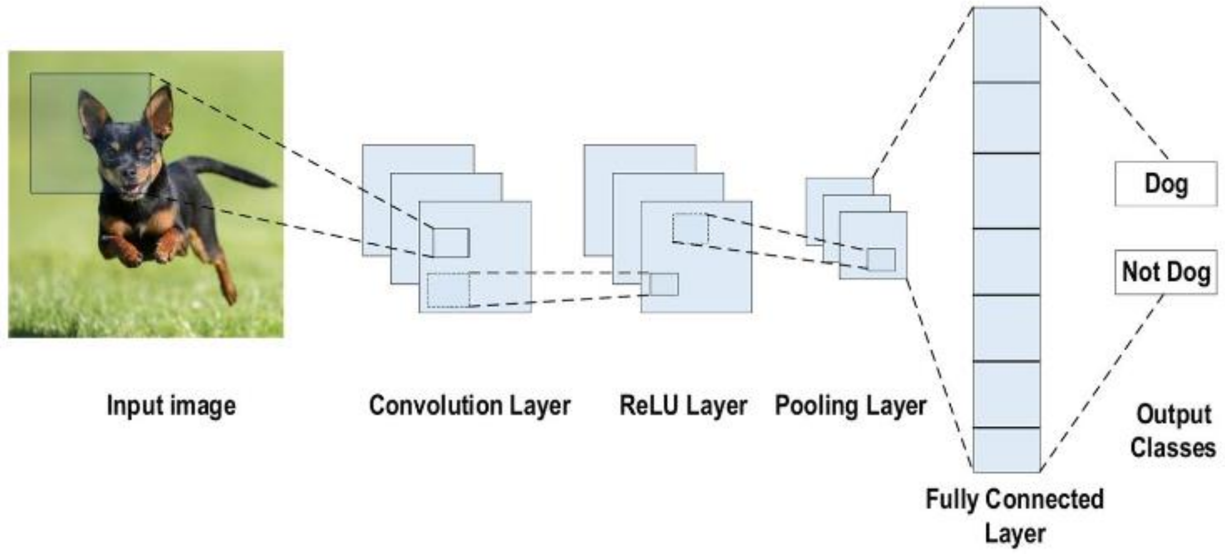


**Figure 1.8** CNN architecture for image classification [30]

## 1.3. Graphical User Interface

Graphical User Interface, or GUI for short, provides users with a visual interface for using software or electrical gadgets. It utilizes graphical elements like icons, buttons, and windows, offering a more intuitive experience compared to text-based interfaces. It enhances user experience by offering an intuitive and easy-to-use environment, allowing users to navigate and operate applications more easily.

In my thesis, I employed a GUI to develop an application with the specific purpose of streamlining the detection of debris after an earthquake. The GUI serves as the interface through which users can interact with the implemented algorithms. This application aims to simplify and enhance the process of identifying and assessing debris objects in post-earthquake scenarios.

# 2. MATHERIAL AND METHODS

## 2.1. Classical image processing algorithms

In the methodology, classical image processing algorithms are used to detect debris post-earthquake from drone-captured images. A dataset of 110 images is utilized, with 10 for testing and 100 to create masks. Various algorithms are applied to select images and manually delineate debris regions to generate masks. Nine effective algorithms are chosen to discriminate debris from other elements in the images, with resulting masks aiding debris identification. These masks are then applied to locate and analyze debris regions, offering a comprehensive approach.

Additionally, a graphical representation will be provided, showcasing the integration of classical algorithms into the graphical user interface (GUI) for debris detection. This graph visually outlines the diverse methods incorporated in the study.

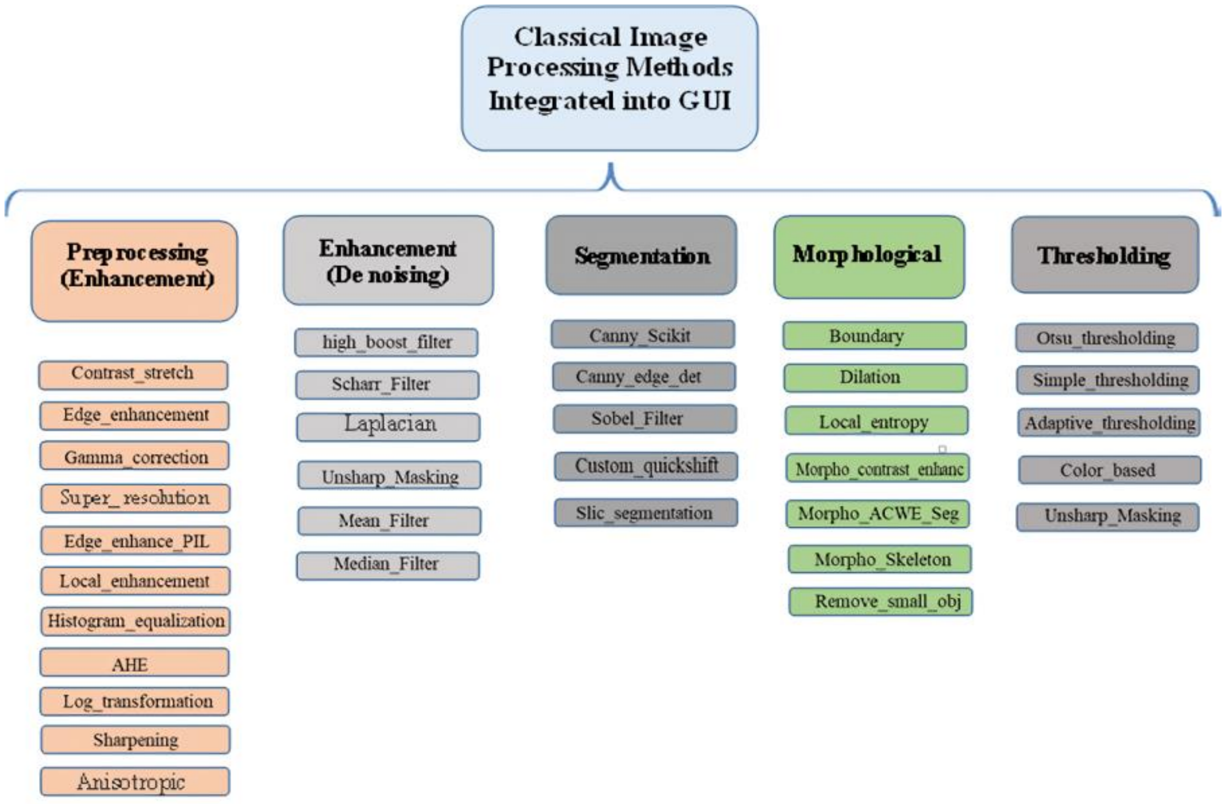


**Figure 2.1** Classical Image Processing Methods Integrated into GUI

### 2.1.1. Preprocessing Techniques for Image Enhancement

In the field of image processing, preprocessing stands out as a pivotal step, allowing for the refinement of raw images to enhance their quality and ease subsequent analysis. This preliminary stage encompasses a range of techniques aimed at improving the visual clarity, removing noise, and enhancing relevant features within the images. One fundamental aspect of preprocessing is image enhancement, which involves the manipulation of pixel values to augment image attributes such as contrast, brightness, and sharpness. By employing various enhancement algorithms, researchers can accentuate important details, mitigate distortions, and optimize images for subsequent processing steps.

*a) Contrast stretch*

The Contrast Enhancement algorithm enhances image clarity by adjusting pixel intensities to amplify contrast between dark and bright areas, thereby improving visual quality and revealing hidden details.

*b) Edge enhancement*

Edge Enhancement algorithm emphasizes edges in an image by boosting the intensity gradients, making them more pronounced and enhancing the visibility of edge features. This technique helps in highlighting object boundaries and structural details, aiding in better analysis and detection tasks.

*c) Gamma Correction*

Gamma correction modifies an image's contrast and hue. by nonlinearly mapping the input intensity values to output values, helping compensate for nonlinear response in displays and improving image quality [16].

*d) Super Resolution*

Super-resolution techniques enhance low-resolution images to reconstruct high-resolution versions, Employing interpolation and learning-based methods to enhance image details, useful in medical imaging and satellite imagery [33].

*e) Edge Enhancement (PIL)*

Edge enhancement is a feature in the Pillow library, a modern fork of the Python Imaging Library (PIL), which enhances the image's ability to distinguish edges. By applying the EDGE_ENHANCE filter from the Image Filter module, Pillow increases the contrast around edges, making them more prominent and detailed. This technique is particularly useful in applications requiring clear edge delineation, such as object recognition and image analysis. With Pillow, edge enhancement can be achieved easily, providing a quick way to improve image sharpness and clarity [5].

*f) Local Enhancement*

Local enhancement techniques selectively enhance specific regions within an image based on local characteristics, preserving global contrast while improving local details, commonly used in medical imaging and satellite imagery analysis [16][34].

*g) Histogram Equalization*

Histogram equalization is a method employed to redistribute pixel intensities within an image, aiming to achieve a uniform histogram. This process enhances contrast and improves the visibility of image details, commonly employed in medical imaging and satellite imagery processing. [16].

*h) Adaptive Histogram Equalization (AHE)*

AHE partitions an image into regions and performs histogram equalization individually on each region, adapting to local image characteristics and enhancing contrast in both bright and dark regions, widely used in medical imaging and remote sensing applications [35].

*i) Logarithm Transformation*

Log transformation adjusts pixel intensities using logarithmic functions, compressing the dynamic range of an image to enhance details in darker areas while preserving highlights, commonly utilized in astronomical imaging and medical diagnostics [16].

*j) Sharpening*

Sharpening techniques enhance image clarity by increasing the contrast along edges, typically achieved through the application of high-pass filters to accentuate high-frequency components, widely employed in microscopy and satellite imagery analysis [16].

*k) Anisotropic Filtering*

Anisotropic filtering reduces image noise while preserving edge features by applying spatially varying filter strengths based on the local gradient magnitude and orientation, commonly used in medical imaging and digital photography [34].

The results depicted in Image 2.2 showcase the diverse effects of preprocessing techniques for image enhancement. While some algorithms were applied to RGB images, others were implemented in grayscale. Despite the already optimal resolution of the input images, the application of these techniques may lead to subtle yet discernible improvements in certain cases. The outputs reflect the nuanced alterations introduced by each algorithm, underscoring their capacity to refine image quality and accentuate features of interest.

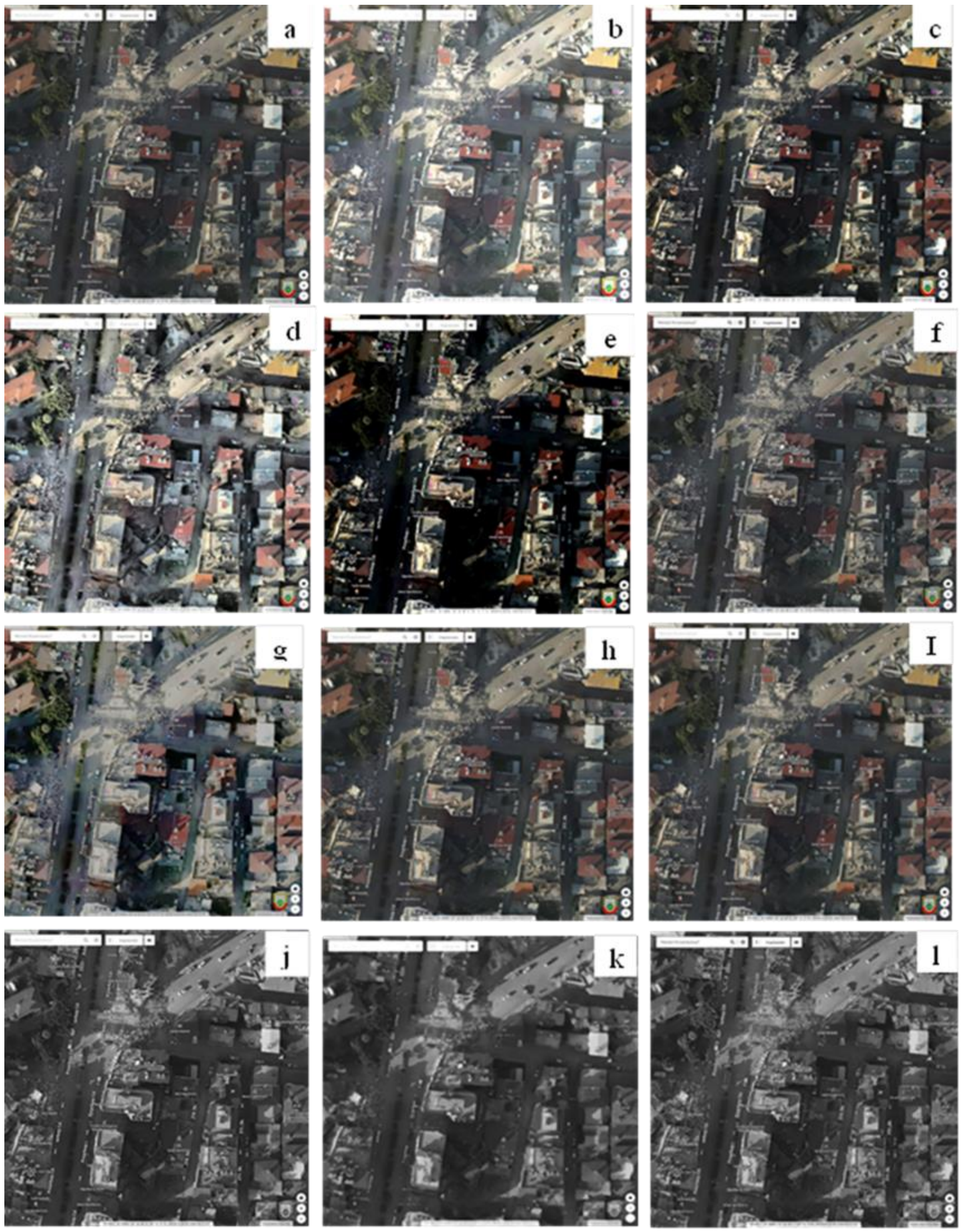


**Figure 2.2** (a)original image (b) Log Transformation, (c) Gamma Correction (d) AHE (e) Contrast stretch (f) Edge Enhancement (PIL) (g) Histogram Equalization (h) Anisotropic Filtering (I) Super Resolution (j) original image(gray scale) (k) Local Enhancement ( l) Edge enhancement(PIL).

### 2.1.2. Denoising Strategies

Denoising plays an essential role in elevating image quality and facilitating analysis by effectively eliminating unwanted noise. High-pass filters emerge as pivotal tools in this process, as they excel in isolating debris by accentuating high-frequency components within the image. Nonetheless, the performance of these filters exhibits variability across different scenarios. For instance, mean and median filters demonstrate minimal alterations in the absence of noise, suggesting their limited efficacy in significantly enhancing image quality under such conditions. Conversely, the Scharr filter, although widely utilized, exhibits constrained effectiveness in combating debris, showcasing the nuanced nature of filter performance in real-world applications. This variability underscores the importance of considering diverse filter options and their specific strengths and limitations in the context of denoising and image enhancement efforts. Additionally, the image in Figure 2.4 illustrates all denoising algorithms, providing visual insight into their application.

1. *High-boost filtering*

High Boost Filtering is a sharpening technique that accentuates high-frequency components in images, effectively enhancing edges and fine details. By amplifying these components, the algorithm sharpens image features and enhances overall clarity [7], [12].

2. *Scharr Filter*

The Scharr Filter, renowned for its gradient-based edge detection capabilities, stands out for its exceptional performance in capturing subtle variations in pixel intensity along edges. This filter excels in accentuating edges within images, facilitating accurate localization and extraction of edge features. By effectively highlighting these edges, the Scharr Filter enhances the overall clarity and detail of images, making it a valuable tool in various image processing tasks. Its ability to discern intricate edge patterns contributes to its widespread use in applications requiring precise edge detection and feature extraction. [36].

3. *Laplacian Filter*

The Laplacian Filter, a sharpening filter utilized in image processing, serves to accentuate edges and intricate details within images. By highlighting regions of rapid intensity change, particularly edges, this filter significantly boosts image contrast and sharpness [32].

In Figure 2.3, a comparison between the original and modified configurations of the Laplacian filter is depicted, elucidating subtle disparities in their impact on image enhancement. Although the conventional filter, featuring a central kernel value of -4, typically proves sufficient, occasional limitations in effectively emphasizing fine details necessitated iterative refinement. Through adjustments to the central kernel value, particularly setting it to -5, notable improvements in edge detection and sharpening capabilities were achieved, underscoring the dynamic nature of algorithmic optimization within the realm of image processing.

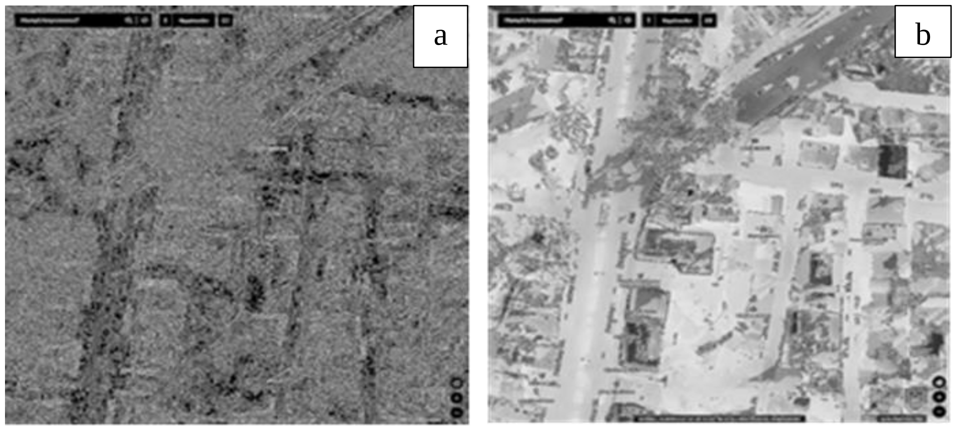


**Figure 2.3.** Comparison of Laplacian Filters: Original Configuration (Image a) vs. Modified with Central Kernel Value -4 (Image b)

*4. Unsharp Masking*

This is a popular technique for sharpening images by accentuating edges and fine details. It involves subtracting a blurred version of the image from the original, thereby enhancing local contrast and increasing image sharpness [17].

*5. Mean Filter*

This filter, also known as the Average Filter, is a spatial filtering technique used for smoothing images and reducing noise. It replaces each pixel's value with the average value of its neighboring pixels, effectively blurring the image and suppressing noise.

*6. Median Filter*

The Filter is a nonlinear filtering technique employed for denoising images by substituting each pixel's value with the median value of the surrounding pixels. Unlike linear filters, the Median Filter preserves edges and fine details while effectively removing impulse noise [32].

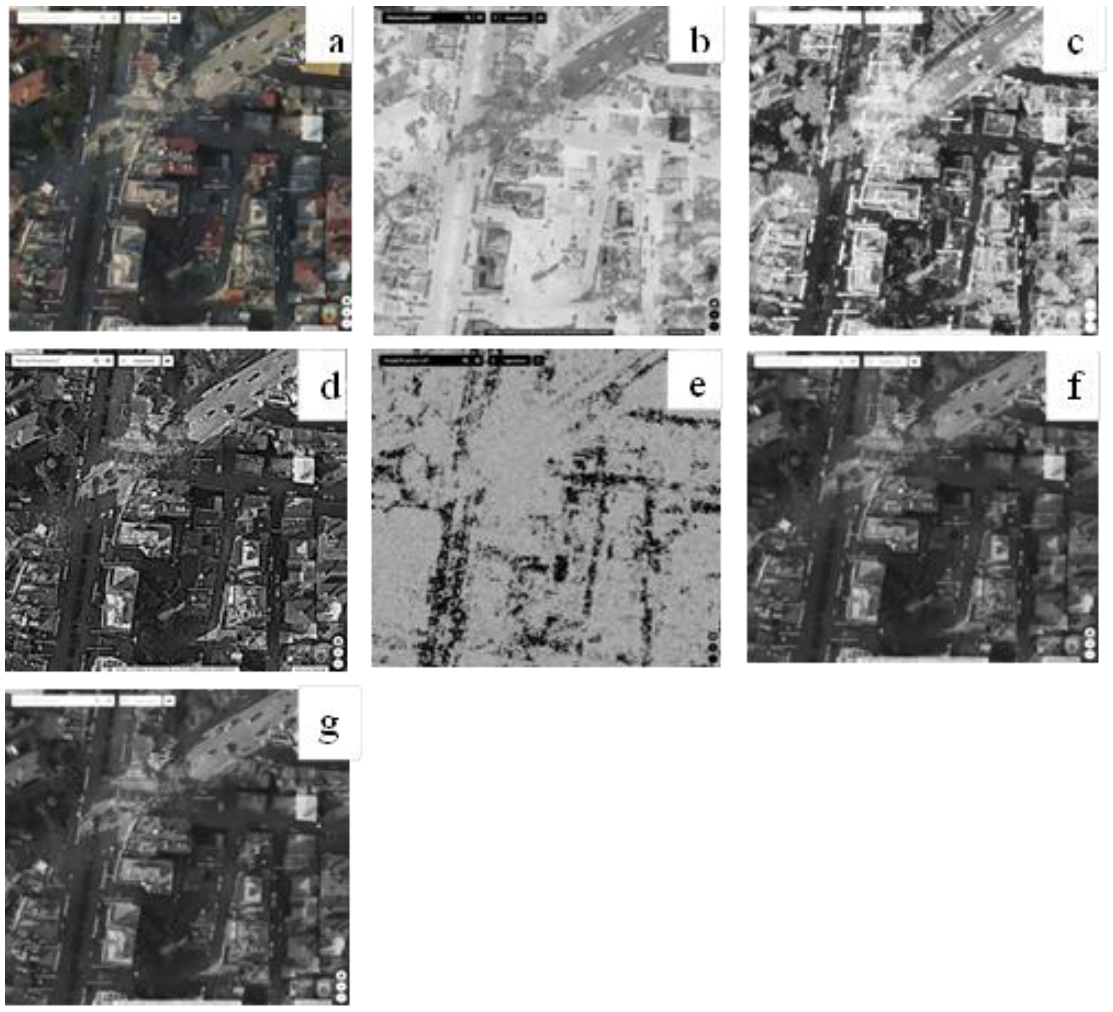


**Figure 2.4** (a) Original image (b) Laplacian Filter, (c) High-boost filtering (d) Unsharp masking (e) Scharr Filter (f) Median filter (g) Mean filter

### 2.1.3. Image Segmentation Techniques

Segmentation is essential in image analysis as it divides an image into separate regions or objects based on specific criteria like color, texture, or intensity. By segmenting an image, we can isolate and identify individual elements within the scene, enabling more targeted analysis and interpretation. In the context of debris detection after an earthquake, Segmentation helps discriminate between debris and other image components by delineating their boundaries. Algorithms like Canny Edge Detection, Sobel Filter, and SLIC Segmentation are employed to identify edges, regions of interest potential debris areas. By segmenting the image into meaningful regions, we can focus our analysis on specific areas of interest, improving the accuracy and efficiency of debris detection algorithms. These Segmentation techniques provide valuable insights into the spatial distribution and characteristics of debris, facilitating effective disaster response and recovery efforts [16] [17].

1. *Canny Edge Detection*

Canny Edge Detection is a popular technique for detecting edges in images. It involves several steps, including noise reduction, gradient calculation, non-maximum suppression, and edge tracking by hysteresis. The algorithm is known for its ability to accurately detect edges while minimizing false positives [32].

2. *Sobel Filter*

The Sobel filter is a widely used method for edge detection in images. By convolving the image with specific kernels, it highlights edges by emphasizing regions of high spatial frequency. Its simplicity and effectiveness make it a popular choice for edge detection tasks in various applications [37].

3. *Custom Quick-Shift Segmentation*

Custom Quick-Shift is an adaptation of the Quick-Shift algorithm, which is a hierarchical clustering algorithm used for image Segmentation. It identifies regions of uniform color or texture in images, effectively segmenting the image into coherent regions [17].

4. *SLIC Segmentation*

SLIC (Simple Linear Iterative Clustering) Segmentation is a super pixel and method for segmenting images that divides them into compact, uniform regions. It groups pixels with similar color and spatial proximity, enabling efficient image Segmentation [38]**.**

The application of Quick-Shift and SLIC Segmentation algorithms for mask creation and their application to test images are elucidated. Firstly, Quick-Shift, a hierarchical Segmentation algorithm, is employed to divide the picture into regions based on color and texture similarity, facilitating the creation of masks for different objects or regions of interest. Subsequently, the SLIC (Simple Linear Iterative Clustering) algorithm is utilized for super pixel Segmentation, generating compact, uniform segments. These segments are then converted into binary masks, serving as templates for isolating specific regions within the images.

The resulting Segmentation algorithms, as depicted in Fig. 2.5, showcase the delineated regions and objects achieved through the utilization of not only Quick-Shift and SLIC algorithms but also Sobel and Canny edge Segmentation methods. This comprehensive approach enhances the efficacy of image Segmentation by incorporating a diverse set of techniques. By integrating Quick-Shift and SLIC for region-based Segmentation along with Sobel and Canny edge detection for edge-based Segmentation, the methodology offers a robust framework for extracting meaningful regions from images. Fig. 2.5 provides a visual representation of the combined effects of these algorithms, demonstrating their collective ability to identify relevant regions of interest and facilitate subsequent analysis or processing.

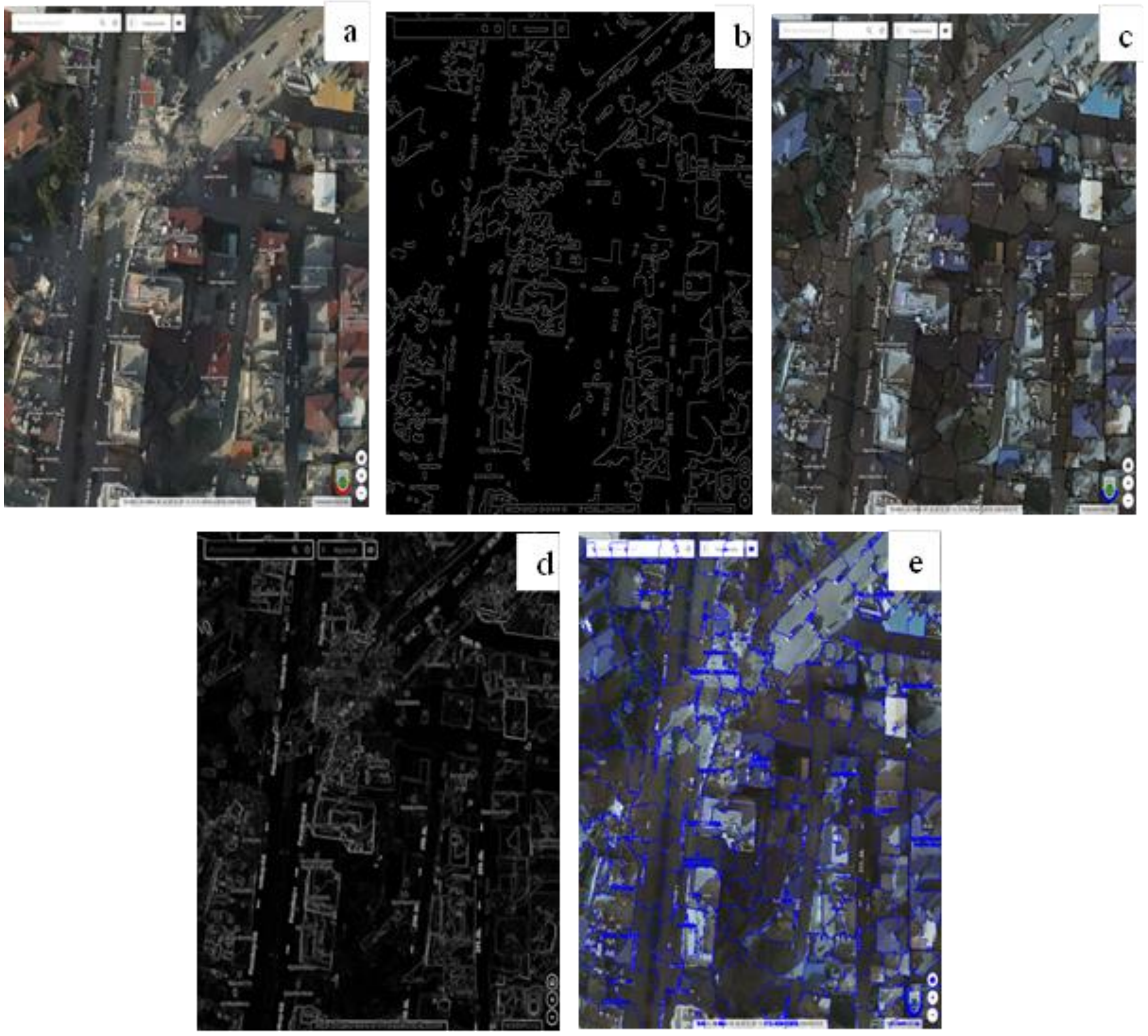


**Figure 2.5** (a) Original image (b) Canny Edge Detection (c) SLIC Segmentation (d) Sobel (e) Custom Quick shift

### 2.1.4. Morphological Operations

Morphological algorithms are essential tools in image processing, providing versatile means to analyze and manipulate image structures. These algorithms operate on binary or grayscale images by probing and modifying pixel values using predefined structuring elements or kernels. Rooted in mathematical morphology principles, these operations include dilation, erosion, opening, closing, and other transformations. Dilation expands image regions, while erosion contracts them. Combining opening and closing operations refines shapes and eliminates noise. Structuring elements define the neighborhood around each pixel, determining the extent of influence during operations. By leveraging these operations, morphological algorithms can extract features, enhance textures, and segment objects within images [37].

In the context of debris detection, morphological operations offer a crucial advantage. They enhance texture features associated with debris, making them more distinguishable from the background. By skillfully applying morphological operations, it becomes easier to isolate debris regions and create masks that highlight these areas for further analysis. These operations contribute significantly to the effectiveness of debris detection algorithms, providing a robust framework for processing images captured in post-earthquake scenarios. Additionally, the image in Figure 2.6 illustrates all morphological algorithms, providing visual insight into their application.

1. *Boundary*

This algorithm extracts the boundaries of objects within an image, highlighting the transition between object and background. It is useful for delineating object shapes and detecting subtle features along their perimeters.

2. *Dilation*

Dilation is a fundamental morphological operation that expands regions in an image, enhancing the presence of features and increasing their size. It's commonly used for filling gaps in objects and joining disjointed components [16].

3. *Local Entropy*

Local entropy-based Segmentation calculates the entropy within local neighborhoods of pixels, allowing for the identification of regions with varying degrees of complexity or uniformity. It can be effective for segmenting objects with heterogeneous textures or structures.

4. *Morphological Contrast Enhancement*

This algorithm enhances the contrast between regions of an image by adjusting the local intensity distribution. It aims to improve the visual appearance of features and facilitate their discrimination from the background.

5. *Morphological ACWE Segmentation*

Active contours without edges (ACWE) Segmentation are a technique that utilizes morphological operations to evolve curves and delineate object boundaries without relying on explicit edge information. It can accurately capture object shapes even in the absence of well-defined edges.

*6. Morphological Skeleton*

The morphological skeleton is a representation of an object that captures its structural characteristics while preserving its topology. It provides a simplified version of the object, useful for shape analysis and pattern recognition tasks.

*7. Remove Small Objects*

This algorithm eliminates small objects or components in an image based on their size or area. It helps to remove noise or irrelevant details, focusing attention on larger, more significant features [17].

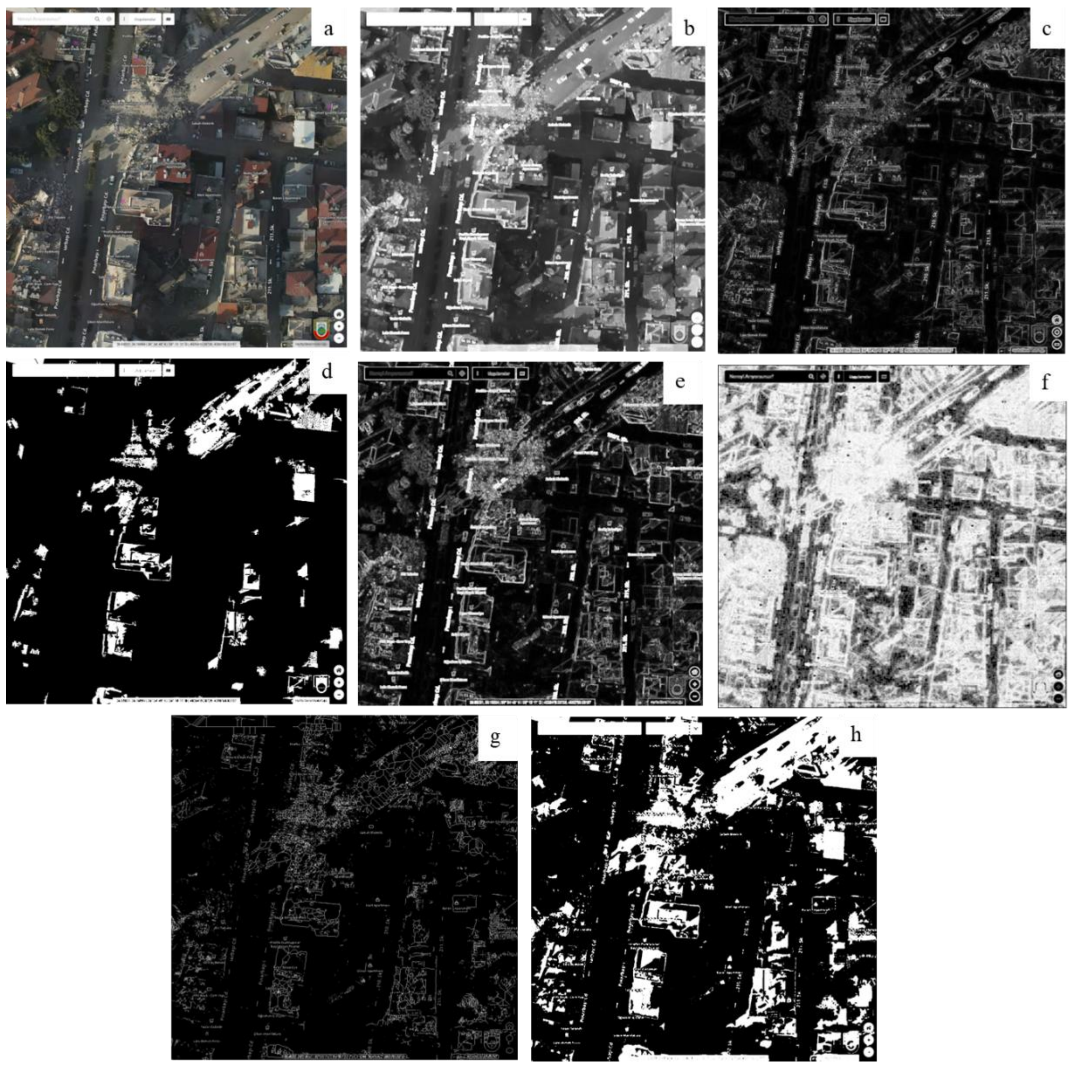


**Figure 2.6** (a) Original image (b) Dilation (c) Boundary (d) Remove Small Objects (e) Morphological Contrast Enhancement (f) Local entropy (g) Skeleton (h) Morphological ACWE

### 2.1.5. Thresholding Methods

Thresholding techniques are essential for segmenting images by dividing them into areas according to the levels of pixel intensity. The outcome of thresholding algorithms is depicted in Figure 2.7. By setting a threshold value, pixels are categorized as foreground or background, allowing the identification of specific features or objects in the image. In the context of detecting debris, thresholding plays a vital role in distinguishing debris from the surrounding background. By choosing an appropriate threshold, regions containing debris can be precisely delineated, facilitating the creation of masks that highlight debris areas for further analysis. This method utilizes intensity variations between debris and the background, enabling effective detection and characterization of debris in post-disaster situations.

1. *Otsu thresholding*

Otsu thresholding is a popular technique for automatically identifying the most suitable threshold value to differentiate between foreground and background pixels in an image. It works by minimizing the intra-class variance of pixel intensities, effectively maximizing the variation in foreground and backdrop between classes. This approach is especially helpful when the histogram of pixel intensities exhibits a bimodal distribution, as it can accurately find the threshold to separate the two classes [40].

2. *Simple thresholding*

Simple thresholding involves selecting a single threshold value to binaries an image, classifying pixels with intensities above the threshold as foreground and those below as background. This method is straightforward and simple to put into practice, yet may not be suitable for images with varying lighting conditions or uneven illumination.

3. *Adaptive thresholding*

Adaptive thresholding dynamically calculates varying threshold values for distinct regions within an image, considering localized variations in illumination. This approach helps overcome the limitations of global thresholding methods, especially in scenarios where the lighting conditions vary across the image [16].

4. *Color based thresholding*

Color thresholding segments images based on specific color characteristics, useful for isolating objects or regions. By setting thresholds for color channels like red, green, and blue, pixels within defined ranges are retained, aiding in object detection and classification.

5. *Unsharp Masking*

Unsharp masking is an image enhancement technique used to sharpen edges and enhance fine details by accentuating high-frequency components. It involves subtracting a blurry version from the original to get a sharper version of the picture, thereby emphasizing edges and edges details [37].

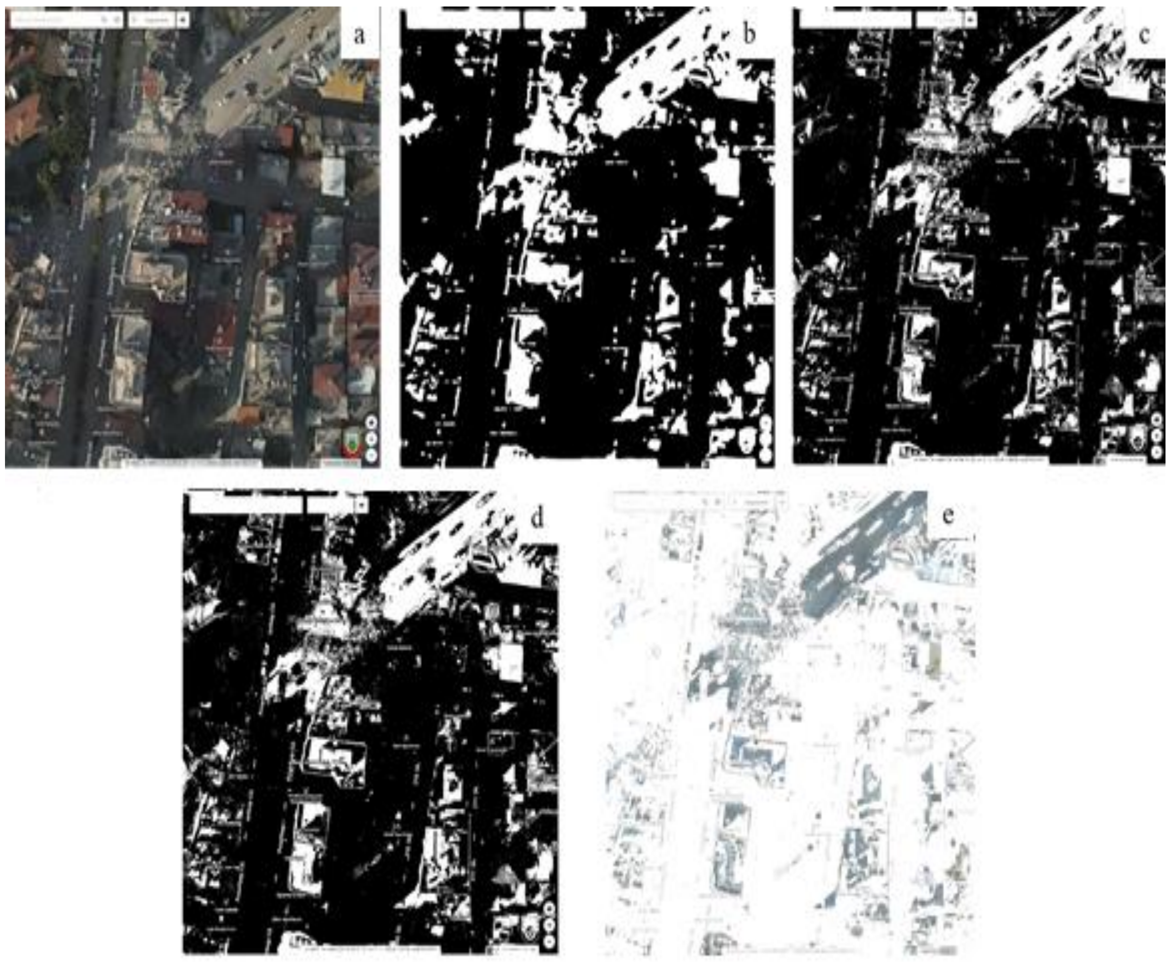


**Figure 2.7** (a) Original image (b) adaptive (c) Otsu (d) Simple thresholding (e) Color based

## 2.2. Exploring Template Matching for Debris Localization

In computer vision, template matching is a method for locating a template picture inside a bigger image. The process involves comparing the template image, which represents the object of interest, to various locations in the larger image. This comparison is typically performed by sliding the template image across the larger image and computing a similarity measure at each point. The position with the maximum similarity score indicates where the template image best aligns with the content of the larger image. Template matching is commonly utilized in operations like object identification, pattern recognition, and image registration. [32].

Correlation, within the framework of template matching, refers to the computation of resemblance between the template picture and different regions of the larger image. The correlation coefficient measures the degree of similarity between two signals or images. In template matching, correlation is used to quantify how well the template image matches each location in the larger image. Higher correlation values indicate stronger resemblance between the template and the image region being examined. However, while correlation can provide a measure of similarity between images, it may not always be efficient for detecting debris or debris texture because of changes in the illumination, image noise, and the complex nature of debris appearance [16].

In this particular example, cross-correlation is employed as a technique by utilizing an eye template image as a kernel, which is then correlated with the raccoon face image. The outcome of this cross-correlation operation provides insights into the precise location of the eye within the raccoon's face image. By employing this process, we can effectively pinpoint and locate the eye region within the broader facial context illustrated in Figure 2.8. This method serves as a powerful tool for detecting specific features, such as the eye, within complex images, enabling detailed analysis and interpretation of facial structures and expressions. Through cross-correlation, we gain useful details on the spatial connection and alignment of features within the image, facilitating various applications in computer vision, image processing, and pattern recognition.

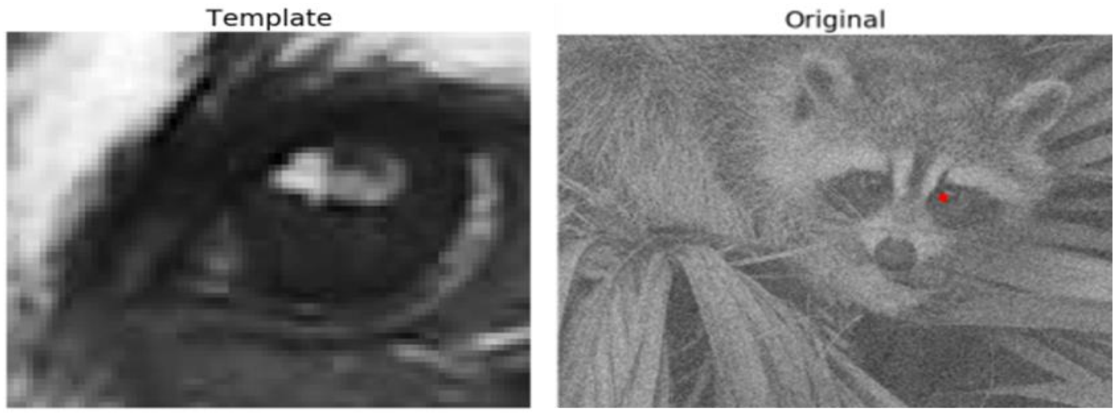


**Figure 2.8** Utilizing Cross-Correlation to Locate Eyes in Raccoon Face Images [16]

## 2.3. Integration of Deep Learning Techniques for Image Analysis

Deep learning techniques, such as object detection and Segmentation, offer significant advantages over classical image processing methods, particularly in scenarios like debris identification after natural disasters. Unlike classical techniques, which often rely on predefined rules and handcrafted features, Deep learning algorithms can autonomously learn and identify pertinent features from the data. This adaptability enables them to effectively handle the complex and diverse characteristics of debris in post-disaster images, including variations in texture, shape, and lighting conditions. Additionally, Models for deep learning such as convolutional neural networks (CNNs), can acquire layered representations of features, enabling them to grasp both fine details and overall meanings of debris objects. By leveraging these capabilities, deep learning-based approaches can achieve excellent output in tasks like debris detection, localization, and Segmentation, making them a compelling choice for automated analysis of disaster-affected areas.

Deep learning methods, distinct from traditional algorithms, necessitate training data to iteratively adjust parameters and extract meaningful representations from input data. This iterative process allows them to discern intricate patterns within images, resulting in more accurate analysis outcomes. Throughout training, Deep learning models fine-tune internal parameters, such as weights and biases, to minimize prediction errors and optimize performance metrics, thus improving generalization to new data. This examination of the training process sheds light on how deep learning algorithms acquire the ability to interpret visual patterns, laying the groundwork for advanced image analysis tasks [32].

### 2.3.1. Custom Data Acquisition and annotation Strategy for Training Deep Learning Models in Debris Detection Post-Earthquake

Training Deep learning models necessitates datasets for several reasons. Firstly, datasets serve as the foundation upon which models acquire the ability to spot patterns and make predictions. In the context of debris detection post-earthquake, datasets provide examples of various types of debris, helping the model learn to distinguish between debris and other objects or backgrounds in images.

Labels are essential components of datasets as they provide ground truth annotations for the data. In the context of debris detection, labels indicate the presence and location of debris within images. These annotations guide the model during training, enabling it to understand the characteristics of debris and learn to identify them accurately.

*a) Finding Datasets:*

Online Platforms: Platforms like Roboflow, Kaggle, and GitHub often host diverse datasets spanning various domains. While these platforms may not offer specific datasets for post-earthquake debris, they serve as starting points for exploration.

Domain-Specific Repositories: Domain-specific repositories and academic databases may contain datasets relevant to disaster response or civil engineering. These repositories can be valuable resources for finding datasets tailored to specific needs.

*b) Custom Dataset Creation:*

In instances where datasets are not readily available on existing platforms, custom dataset creation becomes imperative. This involves collecting and annotating data manually, often through field surveys, drone imaging, satellite images or collaboration with relevant authorities. This tailored approach ensures the dataset aligns closely with the particular demands of the current assignment.

To compile the dataset for training the Deep learning models, satellite imagery from Google Earth Pro and data from the OpenAerialMap site and QGIS were utilized.

1. *Google Earth Pro:* Satellite imagery from Google Earth Pro provided a rich source of visual data. A total of 510 images were gathered from cities in Turkey following the 2023 earthquakes in Turkey and Syria. The majority of these images were captured in Hatay and Gaziantep, providing a diverse dataset for training the models.
2. *OpenAerialMap and QGIS:* To complement the satellite imagery and create a comprehensive dataset, traffic map images were obtained from Open Street Map. QGIS, a geographic information system software, was then used to overlay the traffic map images with the corresponding satellite images captured after the earthquake. This process enabled the creation of paired images showing both the traffic infrastructure and debris distribution post-earthquake.

### 2.3.2. Image Labeling Tools and annotation Process

Labeling involves the process of annotating data with these ground truth annotations. It entails manually or semi-automatically adding labels to images, indicating the presence of debris and specifying its location within the image. This process is crucial for supervised learning tasks, where the model learns from labeled examples to generalize its understanding and make accurate predictions on unseen data.

Roboflow, founded by Joseph Nelson and Brad Dwyer, offers a comprehensive platform for image annotation and dataset preparation. While various methods exist for image annotation, including manual, semi-automatic, and automatic techniques, Roboflow provides an efficient solution for annotation tasks. By stating that 510 satellite images were annotated using Roboflow, readers gain insight into the methodology employed for preparing the dataset. This information underscores the systematic approach taken in labeling the data, contributing to the reliability and robustness of the developed algorithms. Including such details enriches the understanding of the methodology and reinforces the credibility of the research findings.

In addition, Roboflow offers various export versions to cater to different training algorithms and workflows, allowing users to prepare their dataset types according to specific training algorithms such as COCO JSON, YOLO Darknet, YOLOv4, TensorFlow Object Detection (TFRecord), Pascal VOC, Kitti Vision, CreateML, Azure Custom Vision, LabelImg, and MMDetection. These export versions streamline the process of preparing annotated data from Roboflow for specific training algorithms and platforms.

## 2.4. Advanced Object Detection and Segmentation Techniques

### 2.4.1. Object Detection with YOLOv5 and YOLOv8

Object detection in the field of computer vision, represents a fundamental concept extensively explored in scholarly literature. Academic sources delve into the theoretical underpinnings and practical applications of object detection techniques. These texts offer comprehensive discussions on convolutional neural networks (CNNs) and their pivotal role in detecting objects within images. Additionally, scholarly works provide insights into traditional and modern methodologies for object detection, including region-based approaches and cutting-edge Deep learning methodologies. Such literature serves as invaluable resources for researchers, practitioners, and students, fostering a deeper understanding of object detection algorithms, their implementations, and their diverse real-world applications.

Object detection with Deep learning refers to the process of automatically identifying and locating objects within images or videos. Unlike traditional computer vision methods that rely on handcrafted features and algorithms, Deep learning models, particularly (CNNs), derive layered representations straight from raw pixel data. These models can adeptly capture intricate patterns and features, enabling precise localization and classification of objects. [32].

You Only Look Once, or YOLO, is a term for an object detection system that revolutionized the field with its real-time performance and accuracy. Introduced by Joseph Redmon et al. in 2015, YOLO took a unique approach compared to traditional object detection methods via handling it as a regression position to identify spatially separated bounding boxes along with associated class probabilities in one single pass through the neural network. This integrated architecture allowed YOLO to determine bounding boxes and classify objects directly from complete images in real-time, unlike previous methods that relied on region proposal networks [41].

The model outlined in Figure 2 operates by treating object detection as a regression challenge. It segments the input image into an $M \times M$ grid, where each grid cell forecasts N bounding boxes along with their respective confidence scores and K-class probabilities. These forecasts are compiled into a tensor with dimensions $M \times M \times (N * 5 + K)$. Essentially, the model processes the entire image just once, making predictions for each grid cell simultaneously, thus greatly accelerating the detection process. By predicting multiple bounding boxes per grid cell and assigning confidence scores and class probabilities, the model efficiently identifies and classifies objects in the image. Figure 2.9 visually demonstrates this concept, showing the grid structure and the predicted bounding boxes superimposed on the image.

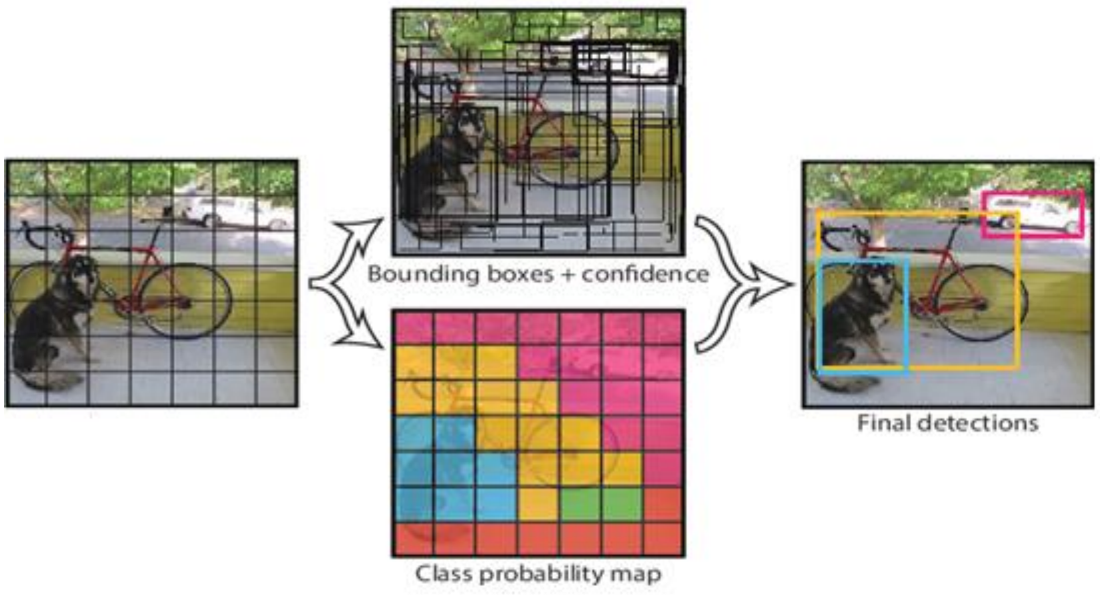


**Figure 2.9** Yolo System Model Detection [41]

The architecture shown in Figure 2.9 details the design of the detection network employed in the YOLO algorithm. It includes 24 convolutional layers crucial for feature extraction from input images. These are followed by 2 fully connected layers that make predictions based on the extracted features. The alternating 1 × 1 convolutional layers significantly minimize the feature space compared to earlier layers, enhancing the network's efficiency. Initially, the network is trained on the ImageNet classification task at a lower resolution of 224 × 224 pixels. However, for detection tasks, the resolution is increased, enabling more detailed analysis and precise object detection [41].

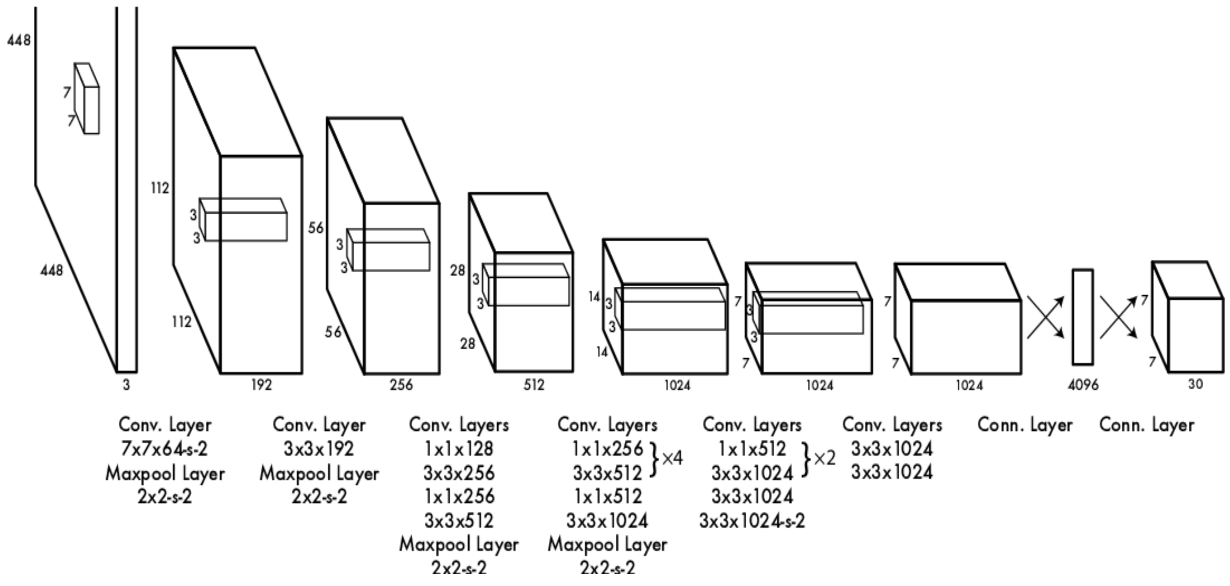


**Figure 2.10** Yolov1 Architecture

In YOLOv5, the model architecture has been divided into three primary parts: the backbone, neck, and head. The backbone utilizes Dark net 53, a novel network architecture emphasizing feature extraction through small filter windows and residual connections. Acting as a bridge between the backbone and head, the neck refines characteristics that the backbone extracts, improving geographical and semantic data at various scales. The head comprises three branches, each with a feature prediction at varying scales and generating bounding boxes, probabilities, and confidence scores. To address overlapping bounding boxes, the network employs Non-maximum Suppression (NMS) [42].

YOLOv8, the latest iteration in object detection model architecture, succeeds YOLOv5 and brings about notable enhancements through a novel neural network structure. Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) are two neural networks that are included in this release. Additionally, an innovative labeling application streamlines the annotation process by offering features like automated labeling, shortcut labeling, and customizable shortcuts, simplifying image annotation for model training.

FPN performs by gradually reducing the pixel density of input images while increasing the quantity of feature channels at the same time. This method generates a feature map capable of detecting objects across various scales and resolutions. Conversely, through skip connections, the PAN design combines functionalities from several network tiers, enabling more effective feature capture across different scales and resolutions. This capability is pivotal for precisely identifying objects in diverse dimensions and forms [42].

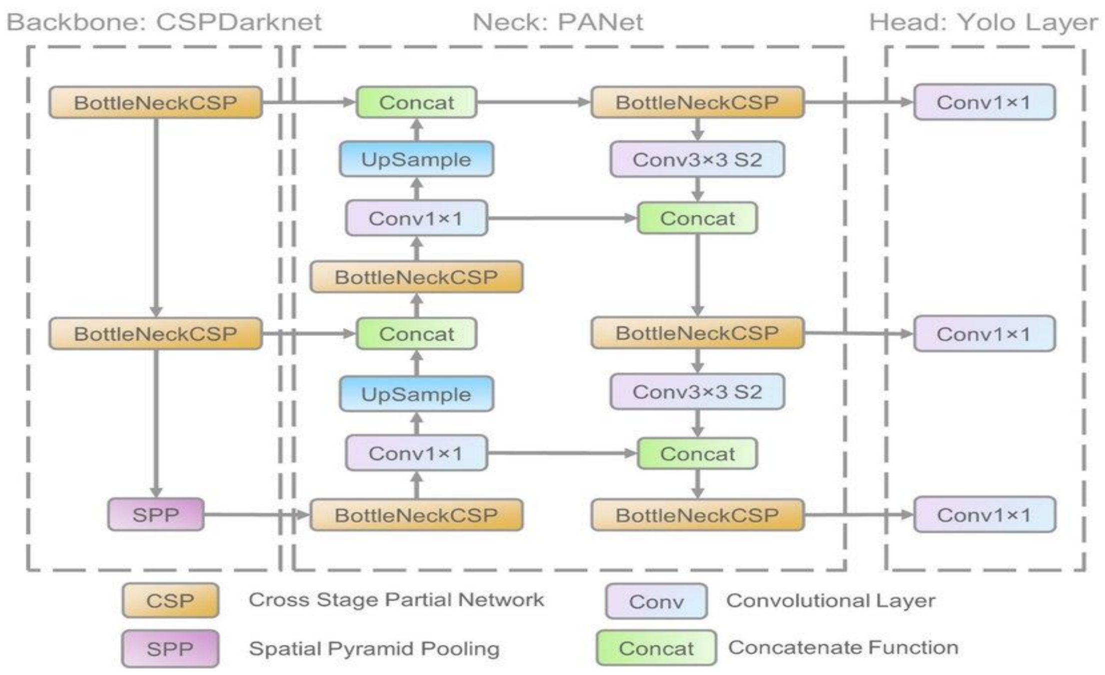


**Figure 2.11** Structure of YOLOv5 [42]

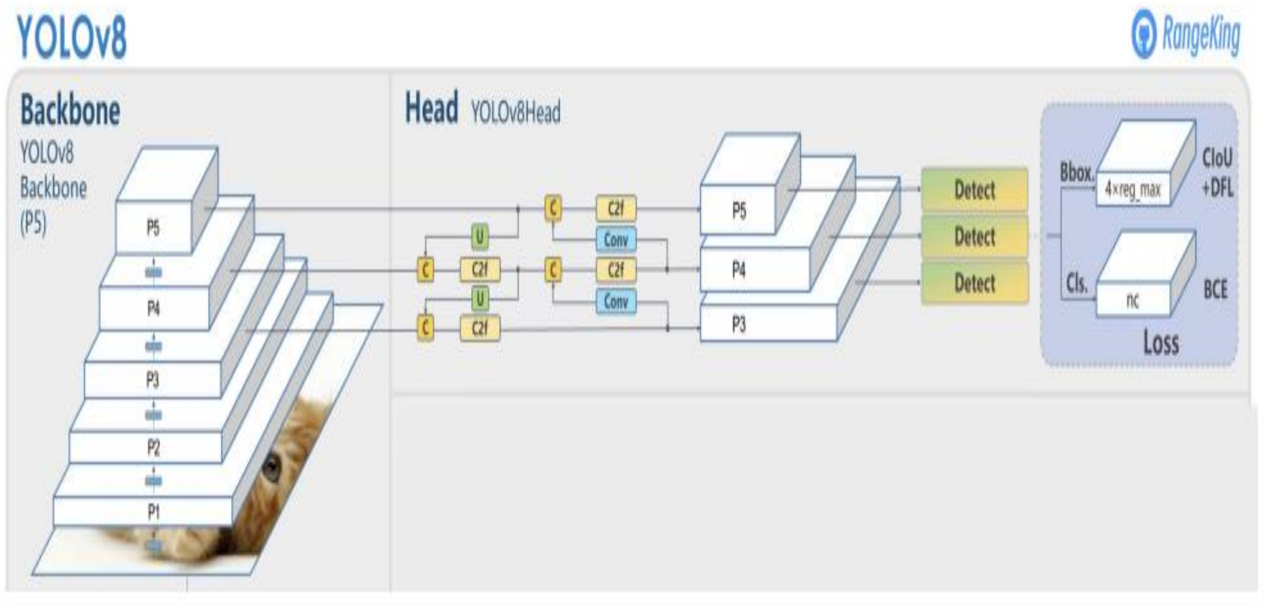


**Figure 2.12** Structure of YOLOv8 [43]

YOLOv8, an advanced iteration of the YOLO object detection model, surpasses YOLOv5 in various aspects, including better mean Average precision (map) and improved performance across different datasets. It introduces a new architecture mixing components from the Path Aggregation Network (PAN) and Feature Pyramid Network (FPN), enhancing feature extraction and accuracy. Moreover, YOLOv8 benefits from training on a more extensive and diverse dataset compared to YOLOv5. It employs a new labeling tool, Roboflow Annotate, for easier image annotation and integrates advanced post-processing techniques like Soft-NMS for refining detection results. Despite being slightly slower, YOLOv8 maintains real-time processing capabilities on modern GPUs. Both YOLOv5 and YOLOv8 utilize mosaic augmentation during training to improve model robustness [43].

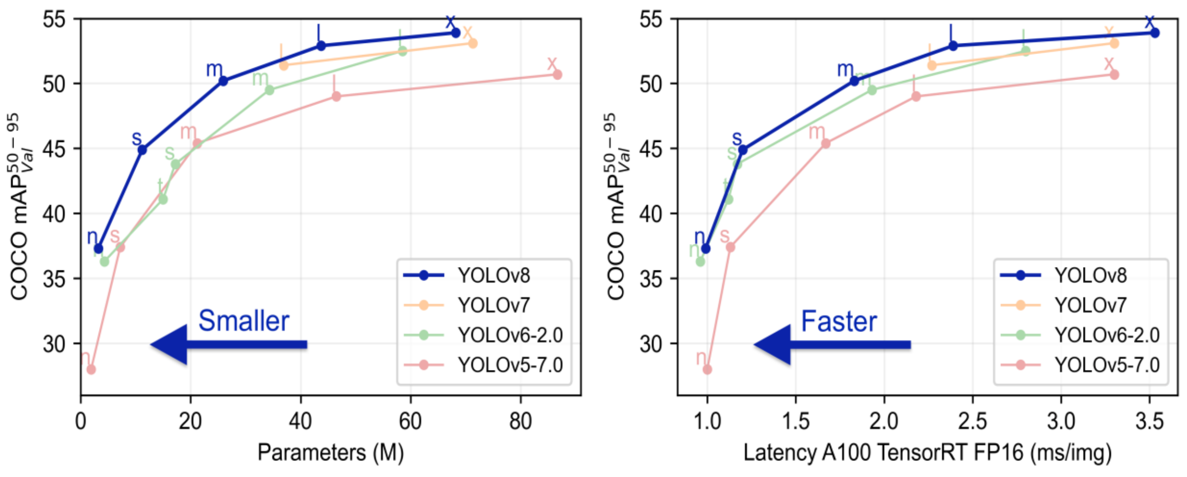


**Figure 2.13** Comparisons of YOLO Versions for Object Detection

[https://github.com/ultralytics/ultralytics?tab=readme-ov-file]

The comparison in Table 2.1 provides an insightful overview of the performance metrics across various iterations of the YOLO (You Only Look Once) models, which include YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. Each model undergoes comprehensive assessment across multiple dimensions, such as its size measured in pixels, the mean Average precision validation score (mAPval), speed metrics on CPU utilizing the ONNX format, speed on the A100 TensorRT platform, Floating Point Operations Per Second (FLOPs) expressed in billions, the parameter count in millions.

This detailed analysis not only offers a comparative examination of these YOLOv8 variants but also facilitates a nuanced understanding of the trade-offs involved in their design. It sheds light on the intricate balance between model size, computational efficiency, and performance characteristics inherent in each variant. Researchers and practitioners alike can leverage this comprehensive dataset to make informed decisions when selecting a suitable model for their specific application requirements. By considering these metrics, they can effectively balance factors such as accuracy, speed, and resource constraints to optimize their choice of YOLOv8 model for their particular use case.

| Model | size (pixels) | $mAP^{val}$ 50-95 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

**Table 2.1** Comparing YOLOv8 Models: Performance Metrics Overview [https://github.com/ultralytics/ultralytics?tab=readme-ov-file]

### 2.4.2 Object Segmentation with Mask R-CNN using Detectron2

The Mask Region-based Convolutional Neural Network, also known as Mask R-CNN, is a sophisticated Deep Learning model that is mostly employed in computer vision applications including object recognition and instance segmentation. Unlike traditional object detection methods that only provide bounding box coordinates, Mask R-CNN goes a step further by not only detecting objects but also precisely outlining their shapes with pixel-level accuracy. This capability makes it invaluable for applications requiring detailed understanding of object boundaries, such as in medical imaging, autonomous driving, and robotics. The architecture of Mask R-CNN consists of a region proposal network (RPN) to generate possible object areas and a backbone (CNN) for feature extraction, and parallel branches for predicting object bounding boxes and Segmentation masks. These components work together to enable Mask R-CNN to identify objects in images while accurately delineating their boundaries, making it a powerful tool for various computer vision tasks [48].

A regional convolutional neural network called Mask R-CNN, involves two primary stages: image scanning and proposal generation, followed by proposal classification, bounding box prediction, and mask generation. This network utilizes the ResNet101 backbone along with Feature Pyramid Networks (FPN) for feature extraction. The (RPN) scans backbone feature maps to efficiently reuse features and minimize redundant computations, producing anchor classes and bounding box refinements. Final proposals undergo further classification and refinement. An ROI pooling algorithm extracts features for classification, leading to the generation of masks for positive regions via fully convoluted layers. Another branch predicts class labels and bounding boxes for each object based on the ROI pooling outputs.

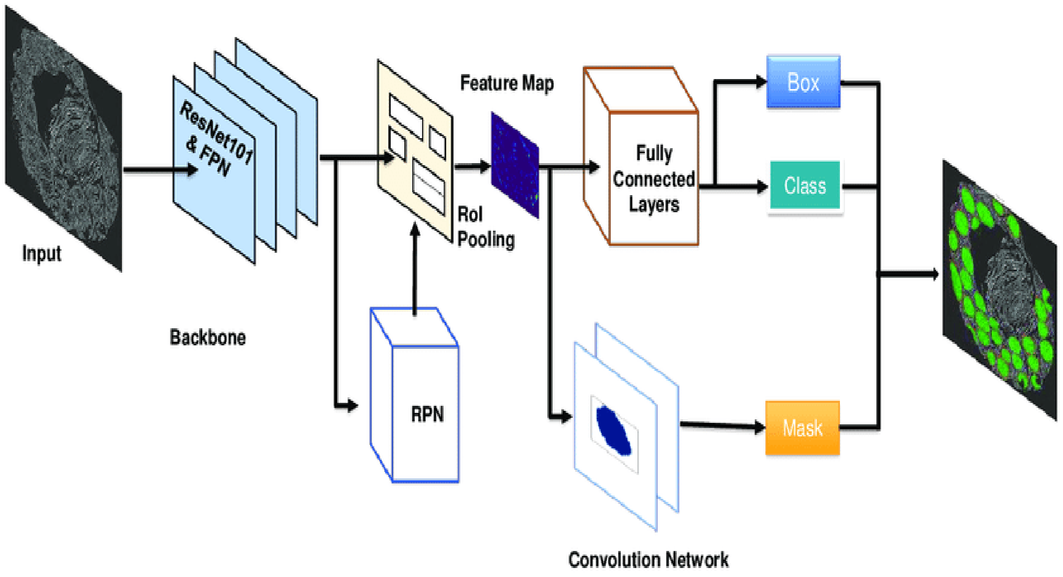


**Figure 2.14** Architecture of Mask R-CNN [49]

Table 2.2 compares the performance metrics of different backbone architectures for Mask R-CNN, the most recent model used for object detection and instance Segmentation. Here's what each column represents:

**Backbone:** Indicates the backbone architecture used for feature extraction.

**AP:** Average precision, a metric that measures the accuracy of object detection.

**AP50:** With an intersection over union (IoU) criterion of 0.5, average precision, which evaluates the precision of object detection with stricter criteria.

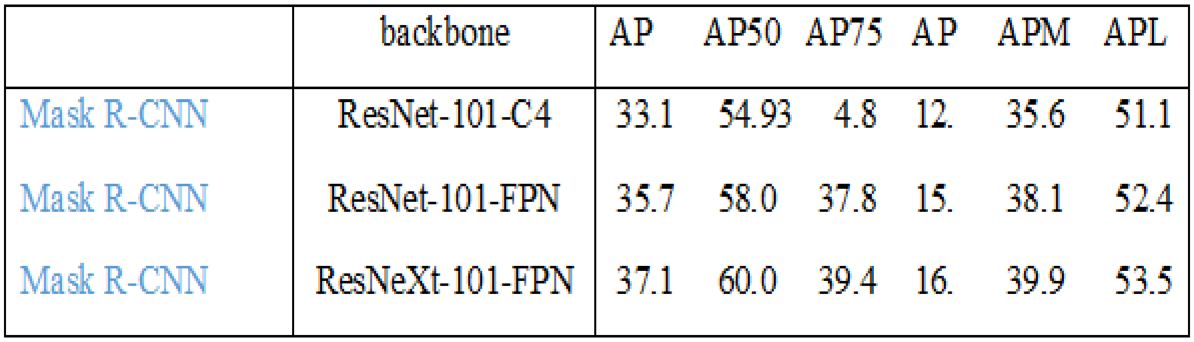
**AP75:** Average precision with a IoU threshold of 0.75, indicating a higher level of accuracy.

**APS:** Average precision for Small objects, focusing on the accuracy of detecting smaller objects.

**APM:** Average precision for Medium objects, assessing the accuracy of detecting medium-sized objects.

**APL:** Average precision for large objects, evaluating the accuracy of detecting larger objects.

The values in the table represent the performance scores achieved by different backbone architectures across these metrics. For example, the higher the AP, AP50, AP75, APS, APM, and APL values, the better the performance of the Mask R-CNN model with the corresponding backbone architecture. In this comparison, the ResNeXt-101-FPN backbone architecture generally outperforms the other architectures in terms of all metrics, indicating its effectiveness in object detection and instance Segmentation tasks [48]



| | backbone | AP | AP50 | AP75 | AP | APM | APL |
|---|---|---|---|---|---|---|---|
| Mask R-CNN | ResNet-101-C4 | 33.1 | 54.93 | 4.8 | 12. | 35.6 | 51.1 |
| Mask R-CNN | ResNet-101-FPN | 35.7 | 58.0 | 37.8 | 15. | 38.1 | 52.4 |
| Mask R-CNN | ResNeXt-101-FPN | 37.1 | 60.0 | 39.4 | 16. | 39.9 | 53.5 |

**Table 2.2** Performance Comparison of Mask R-CNN with Different backbone Architectures [48]

Mask R-CNN can be implemented using Detectron; a popular Deep learning platform created by Facebook Artificial Intelligence Research. Detectron offers a versatile and effective platform for training and deploying object detection models, including Mask R-CNN. By leveraging Detectron's capabilities, developers can easily customize and fine-tune Mask R-CNN models for specific tasks, such as instance Segmentation, object detection, and image Segmentation. Detectron2, developed by Facebook AI Research, represents the latest advancement in libraries for cutting-edge detection and Segmentation algorithms, evolving from its predecessors, Detectron and maskrcnn-benchmark. This platform facilitates various computer vision initiatives and practical implementations within Facebook's research and production environments. To use Mask R-CNN with Detectron2, developers typically follow a series of steps, including data preparation, model configuration, training, and evaluation. Detectron2 offers extensive documentation and tutorials to guide users through each stage of the process, making it accessible to both beginners and experienced practitioners. Implementing Mask R-CNN with Detectron2 allows for seamless integration with other Deep learning techniques and tools, enabling researchers and engineers to build sophisticated computer vision applications with ease [47].

### 2.4.3. Object Segmentation with YOLOv8

In image processing and computer vision, picture segmentation is a fundamental idea, seeking to divide a picture into many areas or parts according to certain criteria. This procedure is essential for a variety of purposes, such object detection, recognition, and scene understanding. In the Deep learning approach, a neural network is often employed to automatically learn and extract features from images, allowing for accurate Segmentation. The Segmentation process includes giving every pixel a label in the image, classifying them into different categories or regions. This makes possible the neural network to understand the spatial distribution of objects within the image and distinguish between different entities. By segmenting images, the Deep learning model can effectively interpret visual data and perform tasks such is crucial for helping machines understand and as identifying objects or delineating boundaries. Overall, image Segmentation plays a vital role in enabling machines to comprehend and interpret in visual understanding, contributing to progress in various domains like medical imaging, autonomous driving, and satellite imagery analysis [44].

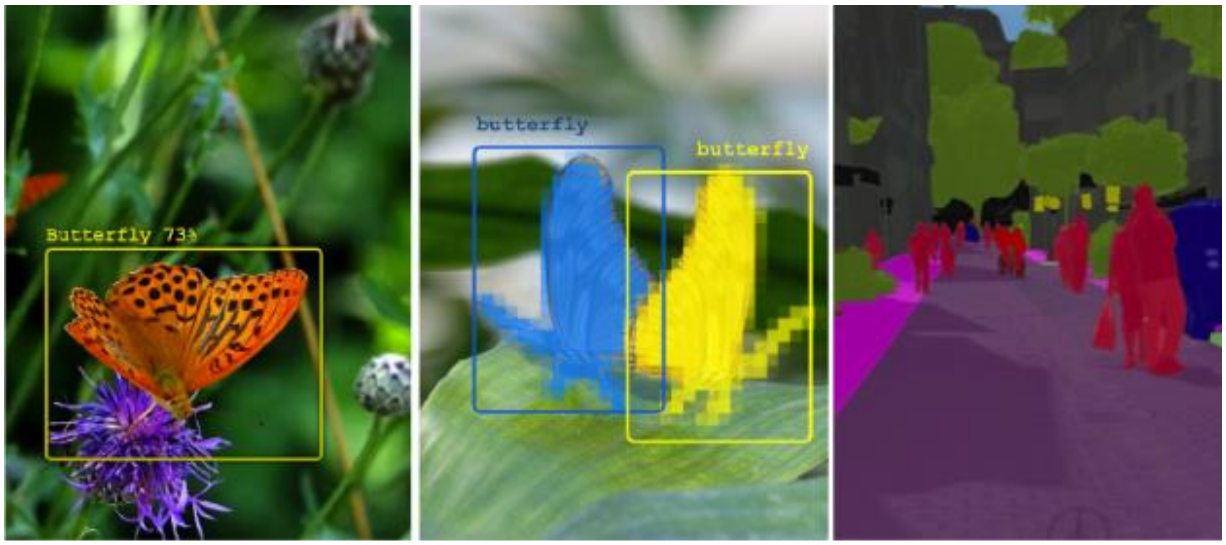


**Figure 2.15** From left to right: object detection, Instance Segmentation, and whole-scene semantic segmentation [44]

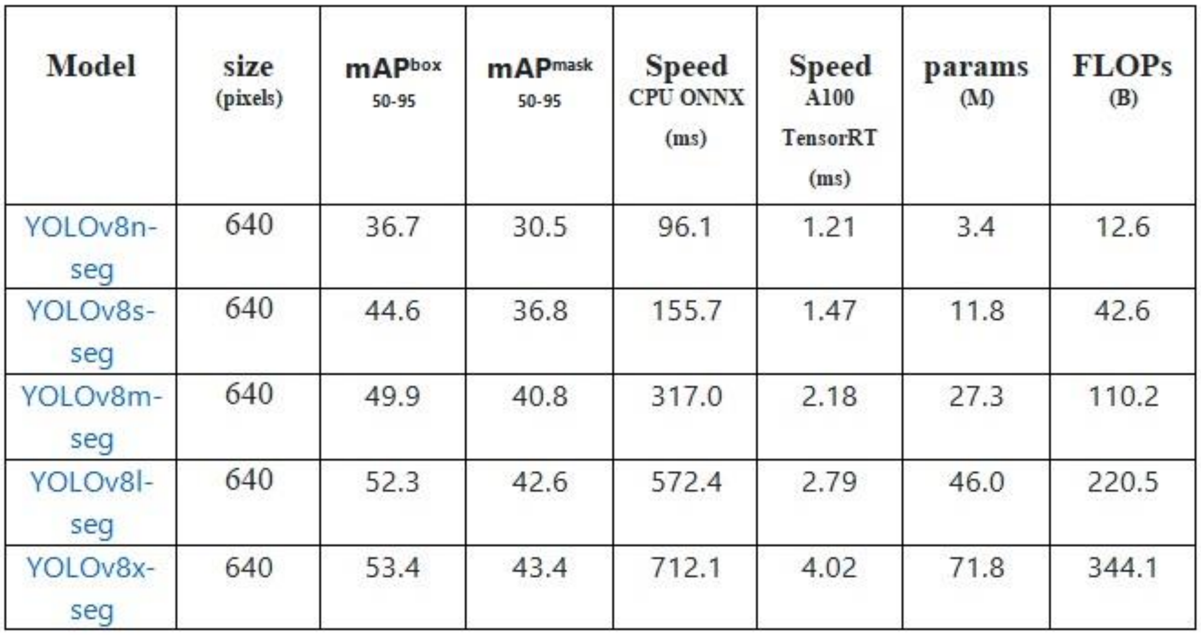
Three different tasks related to computer vision are shown in Figure 2.10: object identification and instance division, and whole-scene semantic Segmentation, showcasing images from the Arthropods and Cityscapes datasets. Object detection entails recognizing and pinpointing specific objects within an image, enabling the recognition of multiple objects simultaneously. Instance Segmentation extends its functionality by not only identifying objects but also distinguishing between individual instances of the same object class, providing pixel-level Segmentation. On the other hand, whole-scene semantic Segmentation aims to assign semantic labels to each pixel in the image, categorizing entire scenes based on their content. Understanding these tasks and their differences is crucial for developing robust computer vision systems capable of handling diverse visual data and real-world applications [44].

YOLOv8 is a cutting-edge object detecting system and Segmentation algorithm that provides real-time performance while maintaining high accuracy. Unlike traditional Segmentation algorithms that require multiple passes over an image, to estimate bounding boxes and class probabilities, YOLOv8 divides the incoming picture into a grid. It then uses this grid to make its predictions. This approach enables YOLOv8 to achieve remarkable speed, making it suitable for tasks requiring immediate response, like self-driving vehicles, surveillance, and robotics.

Compared to other Segmentation algorithms like Mask R-CNN, YOLOv8 offers several advantages. One key advantage is its speed, as YOLOv8 processes images in a single pass, leading to faster inference times. Additionally, YOLOv8 provides instance-level Segmentation, allowing it to differentiate between individual instances of the same object class without the need for additional post-processing steps. However, it may sacrifice some precision compared to more complex Segmentation methods like Mask R-CNN, which perform pixel-level Segmentation. Despite this, YOLOv8' balance between speed and accuracy making it a well-liked option for various computer vision tasks [45].

A comparison of the several YOLOv8 Segmentation models, such as YOLOv8n-seg, YOLOv8s-seg, YOLOv8m-seg, YOLOv8l-seg, and YOLOv8x-seg, is shown in table 2.2. Key metrics such as model size (in pixels), mAPbox (mean Average precision for bounding box detection), mAPmask (mean Average precision for Segmentation mask), speed on CPU using ONNX format (in milliseconds), speed on A100 TensorRT platform (in milliseconds), number of parameters (in millions), and number of FLOPs (Floating Point Operations Per Second, in billions) are provided.

This comparison facilitates an understanding of the performance and computational characteristics across different YOLOv8 Segmentation variants, helping practitioners in deciding which model is best for their particular use case.



| Model | size (pixels) | $mAP^{box}_{50-95}$ | $mAP^{mask}_{50-95}$ | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|---|
| YOLOv8n-seg | 640 | 36.7 | 30.5 | 96.1 | 1.21 | 3.4 | 12.6 |
| YOLOv8s-seg | 640 | 44.6 | 36.8 | 155.7 | 1.47 | 11.8 | 42.6 |
| YOLOv8m-seg | 640 | 49.9 | 40.8 | 317.0 | 2.18 | 27.3 | 110.2 |
| YOLOv8l-seg | 640 | 52.3 | 42.6 | 572.4 | 2.79 | 46.0 | 220.5 |
| YOLOv8x-seg | 640 | 53.4 | 43.4 | 712.1 | 4.02 | 71.8 | 344.1 |

**Table 2.3** Comparing YOLOv8 Models: Performance Metrics Overview [https://github.com/ultralytics/ultralytics?tab=readme-ov-file]

### 2.4.4. Instance Segmentation with Roboflow3

Roboflow is an online platform that simplifies the process of managing and preparing datasets for computer vision tasks, including object Segmentation. It offers a range of tools and features designed to streamline dataset creation, augmentation, and preprocessing. Users can upload their image datasets to Roboflow, where they can perform various operations such as resizing, labeling, and augmenting images to enhance model performance. Additionally, Roboflow provides integration with popular Deep learning frameworks and libraries, allowing users to seamlessly export their prepared datasets in formats compatible with their chosen frameworks. Overall, Roboflow serves as a comprehensive solution for organizing, enhancing, and preparing image datasets for object Segmentation and other computer vision applications.

Upon uploading all 510 satellite images to Roboflow, I leveraged its augmentation methods to expand my dataset to over 1000 images. Through augmentation techniques such as rotation, flipping, and scaling, I increased the diversity and variability of my dataset, thereby enhancing the robustness and generalization ability of my object Segmentation models. Roboflow's efficient augmentation pipeline facilitated the augmentation process, enabling me to efficiently generate a larger and more comprehensive dataset for training and evaluation purposes.

Object Segmentation with Roboflow3 involves utilizing the Roboflow platform to streamline the Segmentation process for images. By uploading images to Roboflow and leveraging its intuitive interface, users can annotate objects within images, defining precise boundaries for Segmentation tasks. Roboflow offers various annotation tools and techniques, such as polygonal Segmentation, to accurately delineate objects of interest. Additionally, users can apply augmentation methods within Roboflow to enhance the dataset, such as rotation, flipping, and color augmentation, thereby improving model robustness. Once annotated and augmented, the dataset can be exported in formats compatible with popular Deep learning frameworks, enabling seamless integration into object Segmentation pipelines. Overall, Roboflow3 simplifies and accelerates the object Segmentation workflow, empowering users to efficiently annotate, augment, and export datasets for training Segmentation models [50].

## 2.5. Challenges and Considerations in Deep Learning Training

a) Overfitting*:* Overfitting arises when a model becomes overly adept at learning from the training data, capturing noise and irrelevant patterns that fail to generalize to new, unseen data.. This can lead to poor performance on new data.

1. High model capacity relative to the dataset size, allowing the model to memorize noise and irrelevant features.
2. Inadequate regularization strategies, such weight decay or dropout, to stop the model from fitting the training set too closely.Lack of diverse training data, leading the model to learn specific examples rather than general patterns.
3. Complex model architectures with many layers or parameters, increasing the risk of overfitting.

b) Underfitting*:* Underfitting happens when a model is too simplistic to recognize the underlying patterns in the data. It thus does badly on the training and test datasets.

1. Insufficient model capacity to capture the complexity of the underlying data distribution.
2. Inappropriate choice of model architecture or complexity for the given task.
3. Limited training data or inadequate representation of different classes or categories in the dataset.

c) Data Imbalance*:*Data imbalance occurs when one class or category dominates the dataset, leading to biased model predictions and reduced performance on minority classes.

1. Skewed distribution of classes or categories in the dataset, resulting in biased model predictions.
2. Insufficient samples for minority classes, making it challenging for the model to learn their representations effectively.
3. Inadequate data preprocessing techniques to address class imbalances, such as data augmentation or resampling.

d) Vanishing and Exploding Gradients:In deep neural networks : it occur when the gradients become too small during backpropagation, hindering learning. Conversely, exploding gradients occur when the gradients become too large, causing instability in training.

   1. Deep networks with many layers can suffer from vanishing gradients, especially during training with activation functions like sigmoid or tanh.
   2. Poor weight initialization strategies, leading to gradients that either vanish or explode as they propagate through the network.
   3. Lack of normalization techniques, such as batch normalization, to stabilize gradients and mitigate exploding or vanishing gradient issues.

e) Local Minima and Plateaus*:*Deep learning optimization landscapes often contain numerous local minima and plateaus, making it challenging for optimization algorithms to converge to the global optimum.

   1. Non-convex nature of the optimization landscape in Deep learning, resulting in multiple local minima and plateaus that can trap optimization algorithms.
   2. High-dimensional parameter space, making it difficult for optimization algorithms to escape local minima and reach the global optimum.

f) Learning Rate Decay*:*Inappropriate learning rate schedules or decay rates can slow down or hinder the convergence of the model during training, affecting its ability to learn effectively.

   1. Inappropriate learning rate schedules or decay rates that cause the learning rate to decrease too quickly or too slowly during training.
   2. Failure to adjust the learning rate based on the training dynamics or convergence behavior of the model.

g) Batch Size Selection*:*Improper selection of batch size can impact the stability and convergence of the training process. A batch size that is too small might introduce noise, whereas a batch size that is too big could cause poor generalization or sluggish convergence.

   1. Small batch sizes may introduce noise and instability in the training process, affecting gradient estimation and convergence.
   2. Large batch sizes can lead to slower convergence or poorer generalization due to reduced stochasticity in the optimization process.

h) Model Complexity:Deep learning models with excessive complexity relative to the dataset size are prone to overfitting, as they may memorize training examples instead of learning meaningful patterns.

   1. Overly complex models with excessive parameters relative to the dataset size may memorize training examples rather than learning meaningful patterns.
   2. Lack of regularization techniques to constrain the model's complexity and prevent overfitting.

i) Validation Set Leakage*:* Information from the validation set being leaked into the training process could result in overfitting. It's crucial to ensure that the validation set remains independent and unseen during model training.

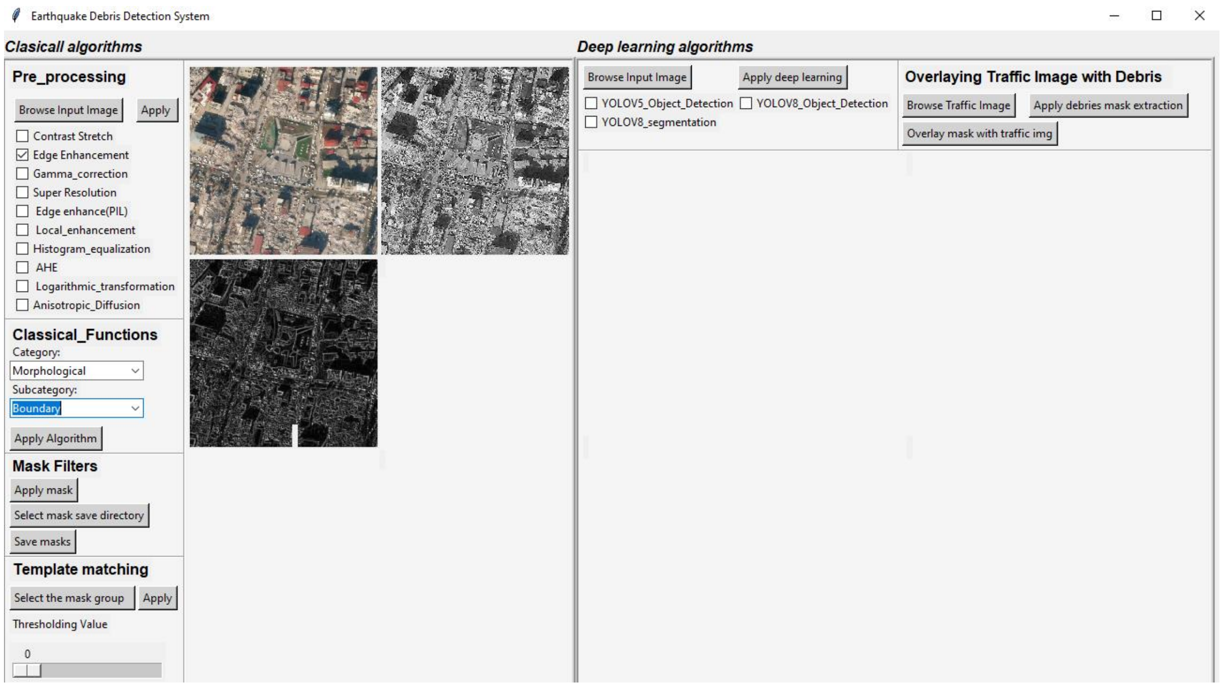
   1. Unintentional contamination of the validation set with training data, leading to overly optimistic performance estimates during model evaluation.
   2. Improper handling of data preprocessing steps or feature transformations that inadvertently leak information from the validation set into the training process.

# 3. RESULTS AND DESCUTIONS

## 3.1. Classical Image Processing Results

I developed a user-friendly Graphical User Interface (GUI) that combines classical and deep learning algorithms, each serving specific roles. The classical algorithms allow users to apply various image processing techniques, visualize outcomes, and use masks for further analysis. Additionally, the Deep learning aspect assists in identifying debris post-earthquake.

For classical image processing, I applied a range of traditional algorithms to a randomly selected dataset. Details of these algorithm outcomes are provided in the materials and methods section. Figure 2.2 demonstrates the process of selecting input images to apply different preprocessing and classical algorithms within the GUI. This streamlined approach enables users to easily apply algorithms and visualize output images, providing insights into potential image changes induced by the algorithms.



- ☐ Contrast Stretch
- ☑ Edge Enhancement
- ☐ Gamma_correction
- ☐ Super Resolution
- ☐ Edge enhance(PIL)
- ☐ Local_enhancement
- ☐ Histogram_equalization
- ☐ AHE
- ☐ Logarithmic_transformation
- ☐ Anisotropic_Diffusion

Category: Morphological
Subcategory: Boundary

**Figure 3.1** Pre-processing and Classical Algorithm GUI

Moving forward, in the results and discussion sections, I aim to delve deeper into the methodology of template matching with correlation. By assessing the correlation between the templates and overlapping regions of the image, template matching enables the detection of specific features or patterns. In this study, template matching with correlation was applied to the dataset, and its effectiveness in achieving the research objectives will be thoroughly examined and discussed in subsequent sections.

Given the complexity of identifying debris, I initiated the exploration by employing template matching. This involved selecting a mask from an original image and applying it to ensure the functionality of the template matching approach. This preliminary step was crucial in validating the efficacy of the template matching technique. Subsequently, I proceeded with further investigation, building upon the initial findings to delve deeper into the process of debris detection.

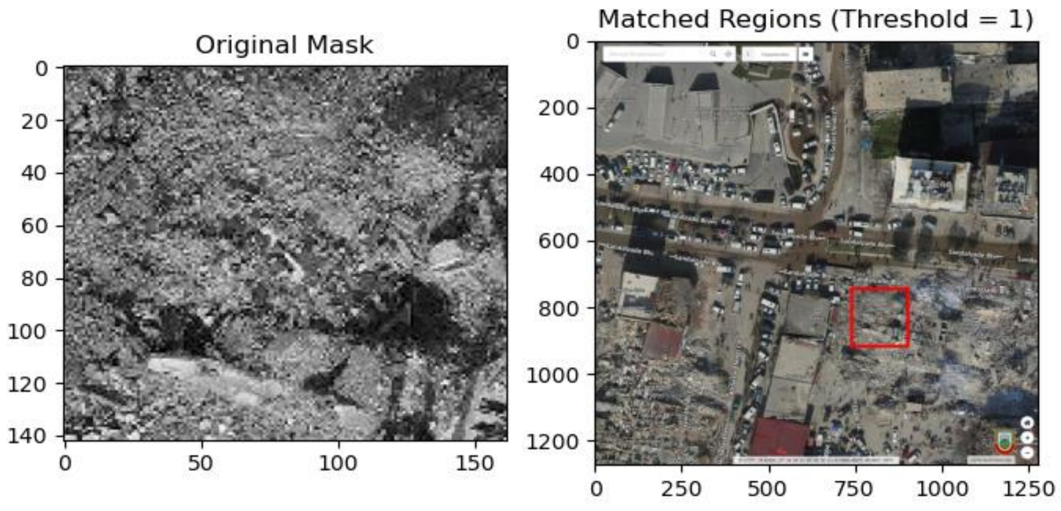


**Figure 3.2** From left to right: original mask and input image

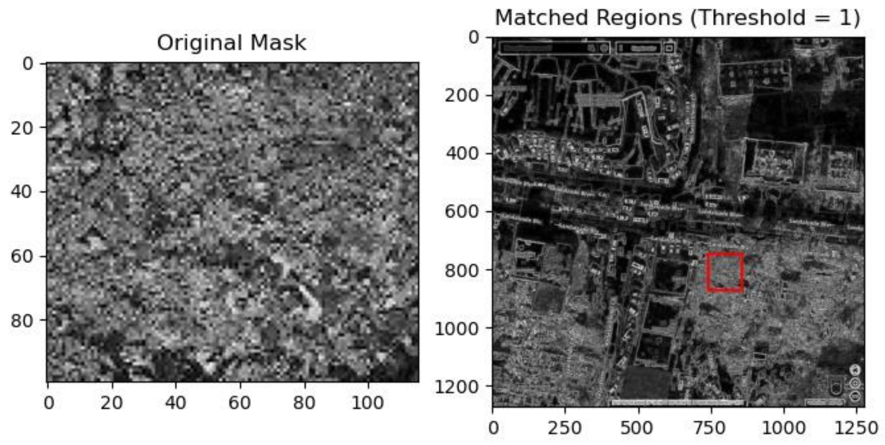


**Figure 3.3** From left to right: the original mask and an image processed with the boundary algorithm

The outcomes show the efficacy of employing the template matching with correlation method. By specifically choosing the mask from the input image, the process ensures that the selected template aligns seamlessly with the corresponding features present in the input image. This meticulous approach not only validates the accuracy of the template matching technique but also ensures that the identified mask precisely corresponds to the desired features within the input images. Additionally, the region field, configured with a threshold value set to one, further corroborates the effectiveness of the template matching process. This setting signifies that the identified mask perfectly matches the features in the input images without any deviations, thus affirming the robustness and reliability of the template matching methodology in refining and fine-tuning the masks to achieve optimal alignment with the target features.

In the subsequent examination, I applied various classic image processing algorithms and singled out those that excel in discriminating debris textures more effectively than others. These selected algorithms encompass boundary detection, color-based thresholding, high-boost filtering, Laplacian filtering, local entropy calculation, Otsu thresholding, Sobel filtering, and unsharp masking. The selection process was meticulous, focusing on their proven effectiveness in capturing nuanced texture variations indicative of debris. Although the exact count of generated masks may vary for each algorithm, the average number remains around 100, with slight deviations observed in some cases. These masks, each measuring 50x50 pixels, were derived from the output of the respective algorithms applied to the test dataset. This comprehensive approach ensures thorough evaluation across diverse textures and environmental conditions, facilitating a comprehensive comparative analysis to pinpoint the most suitable technique for precise debris discrimination and Segmentation. Below, you will find the collection of masks generated by each algorithm, along with the results of template matching using correlation for each algorithm.

We recognize that classic image algorithms may not possess the same level of power and versatility as Deep learning methods. However, the primary objective here is not to showcase the superiority of classic image algorithms but rather to illustrate their functionality specifically in the context of debris detection. This comparative analysis allows us to observe how these traditional methods perform in a targeted application area, providing valuable insights into their effectiveness and potential limitations.

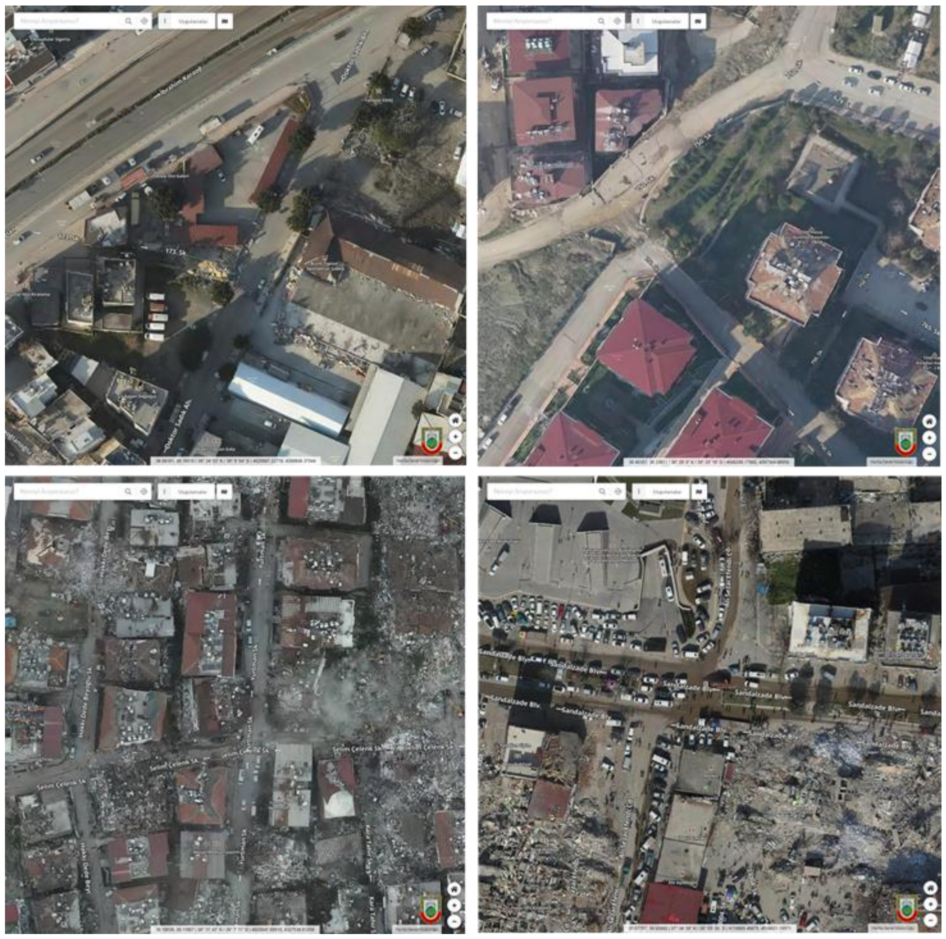


**Figure 3.4** Test set images chosen for region matching analysis.

In this segment, a variety of numerical algorithms were employed to evaluate their performance in template matching with specific masks. This comparative analysis serves two primary objectives: firstly, it enables a comprehensive assessment of each algorithm's capability to detect and precisely match predetermined templates within the input images. Secondly, it offers insights into the effectiveness of these algorithms for template matching, helping to identify which ones are more suitable for this purpose. This methodology enhances our comprehension of the strengths and limitations of each algorithm, thereby aiding in informed decision-making concerning their suitability and effectiveness in template matching tasks.

In this segment, a variety of numerical algorithms were employed to evaluate their performance in template matching with specific masks. This comparative analysis serves two primary objectives: First, it enables a comprehensive assessment of each algorithm's capability to detect and precisely match predetermined templates within the input images. Secondly, it offers insights into the effectiveness of these algorithms for template matching, helping to identify which ones are more suitable for this purpose. This methodology enhances our comprehension of the strengths and limitations of each algorithm, thereby aiding in informed decision-making concerning their suitability and effectiveness in template matching tasks. Additionally, Figure 3.3 comprises four images, two of which depict debris, while the remaining one illustrates a different object. These images were selected from the test set, and they were not utilized in mask creation, serving solely for visual representation and comparative analysis purposes.

It's notable that, in certain algorithms, no regions meet the specified threshold value. Conversely, in others, multiple regions are identified, although many may not pertain to debris. This variability arises due to the diverse textures and lighting conditions present in the debris images, including instances where debris is obscured by shadows or exhibits varying degrees of brightness and texture. Consequently, the outcomes of classical algorithms for debris detection may not consistently meet expectations, reflecting the nuanced challenges inherent in this task.

The subsequent section presents the masks generated by each algorithm alongside their respective results. It becomes evident that the boundary detection algorithm outperforms the others, yielding more satisfactory outcomes. Conversely, the performance of the remaining algorithms appears less robust, indicating room for improvement or potentially rendering them unsuitable for template matching purposes. This observation underscores the need for further refinement and optimization of these algorithms to enhance their efficiency and efficacy in template matching applications.

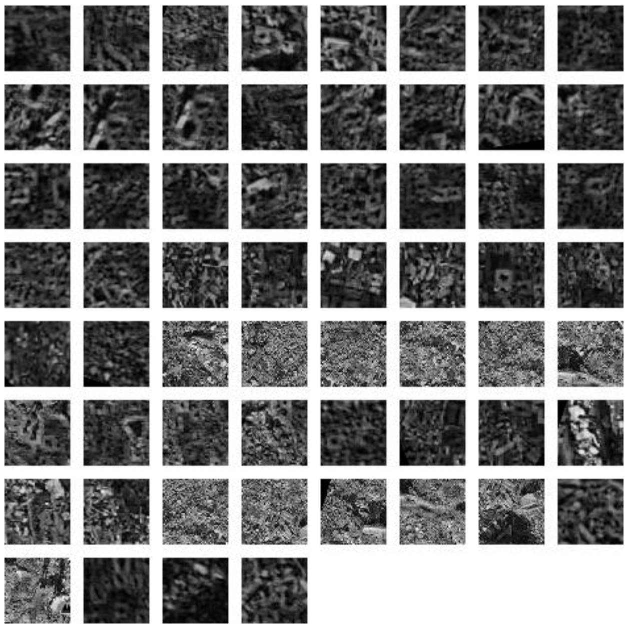


**Figure 3.5** Composite Masks of Debris Regions Identified by Boundary Algorithm Post-Earthquake.

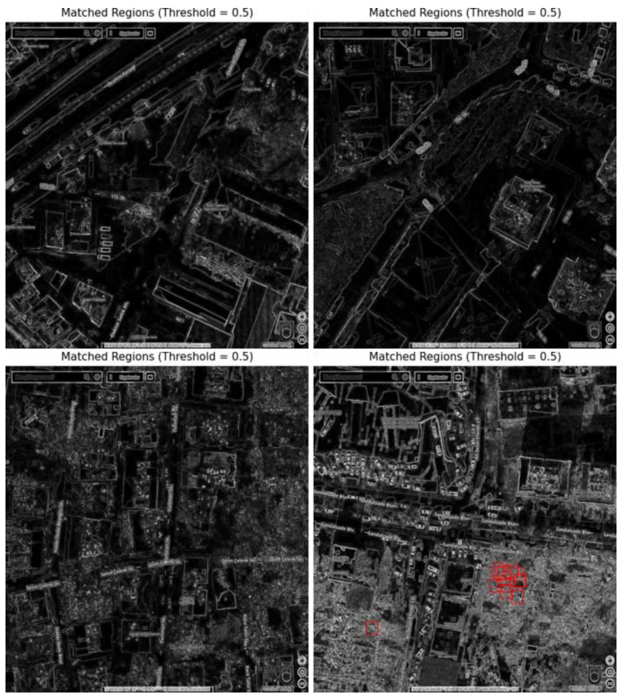

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.6** Matched Regions Detected by Boundary Masks (Threshold 0.5.)

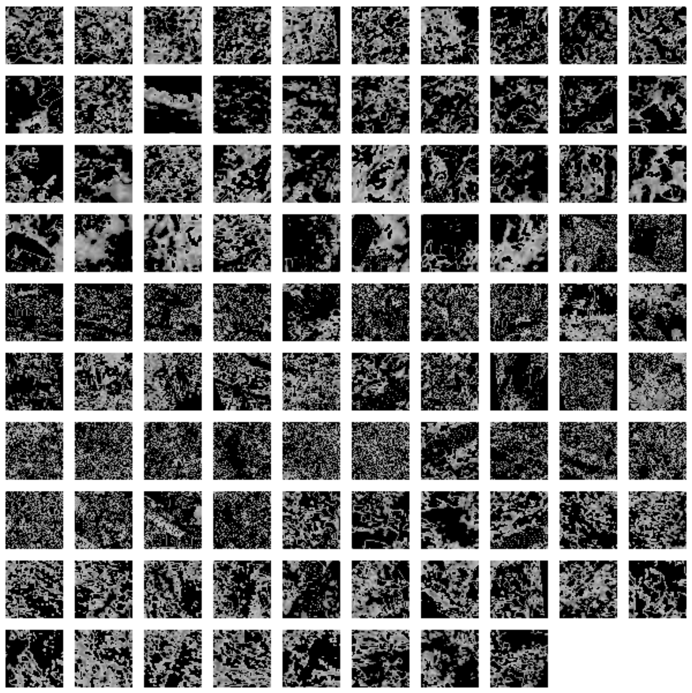


**Figure 3.7** Composite Mask of Debris Regions Identified by color_threshold Algorithm Post-Earthquake

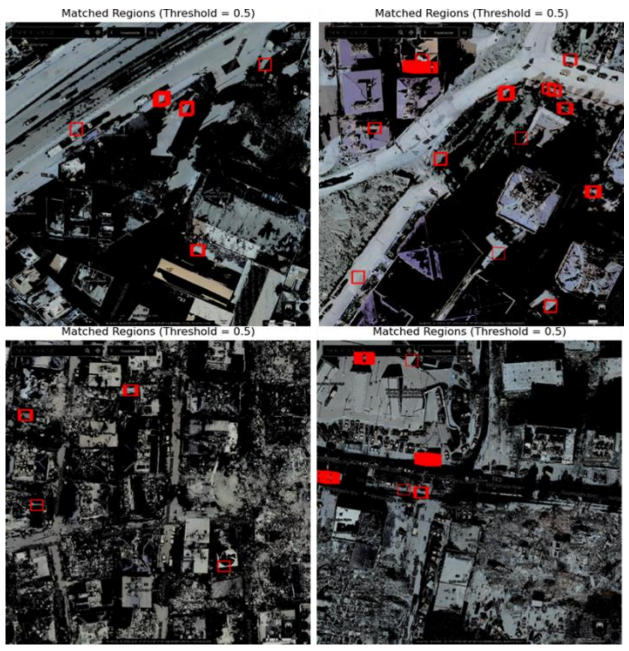

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.8** Matched Regions Detected by color based thresholding Masks (Threshold 0.5)

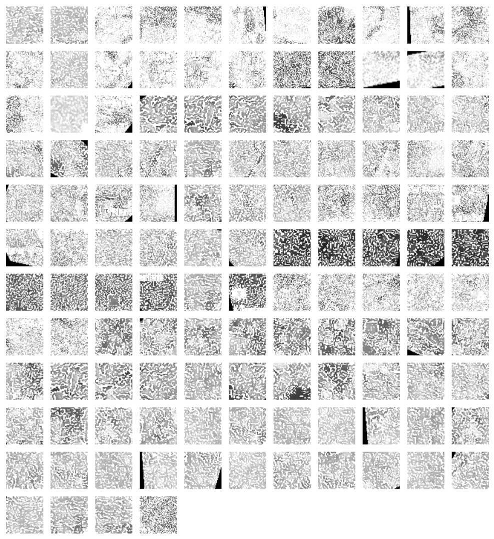


**Figure 3.9** Composite Mask of Debris Regions Identified by high boosts Algorithm Post-Earthquake

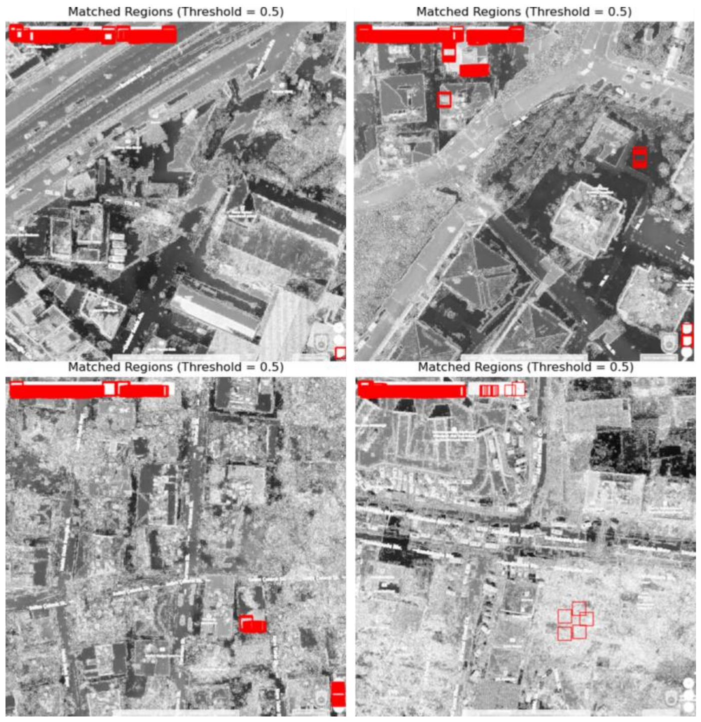

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.10** Matched Regions Detected by color based high boost masks (Threshold 0.5)

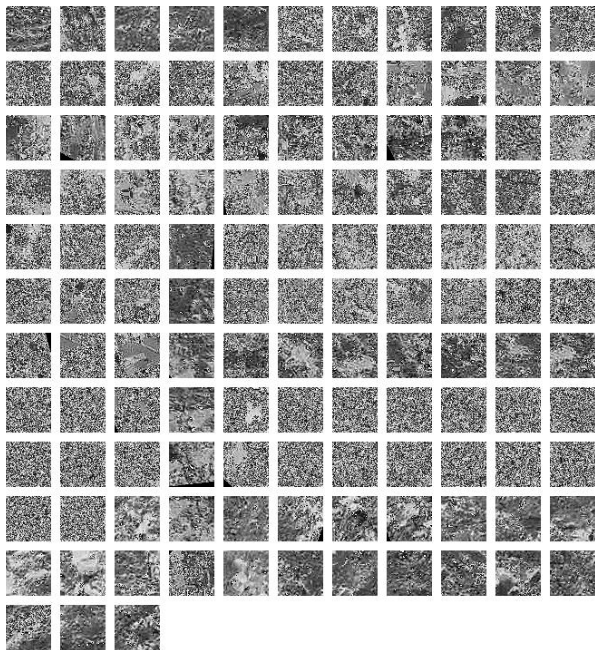


**Figure 3.11** Composite Mask of Debris Regions Identified by laplacian Algorithm Post-Earthquake

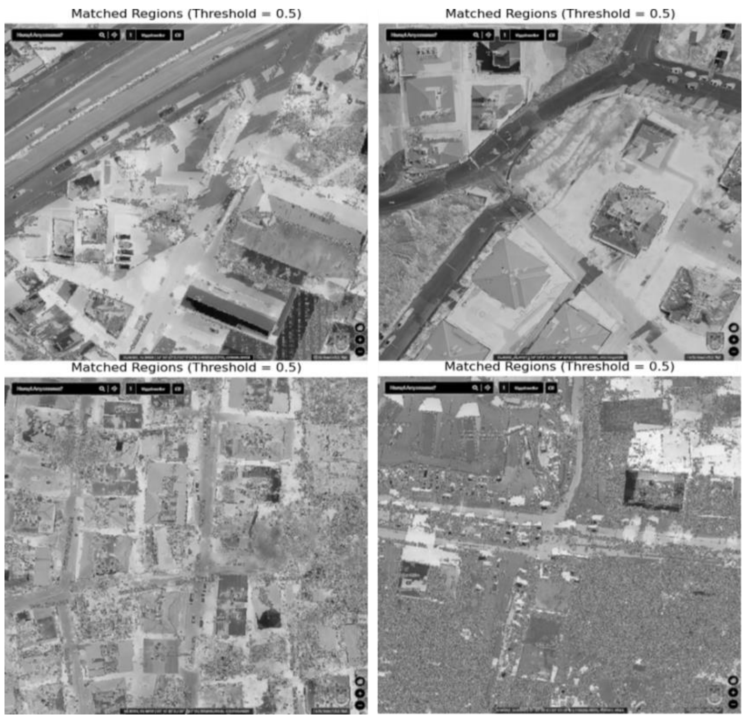

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.12** Matched Regions Detected by color based laplacian Masks (Threshold 0.5)

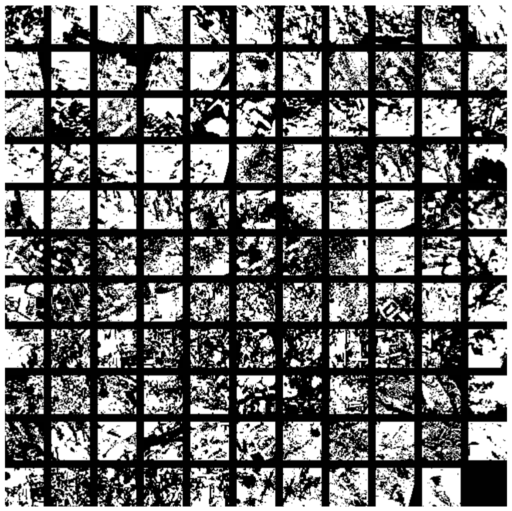


**Figure 3.13** Composite Mask of Debris Regions Identified by odsu thresholding Algorithm Post-Earthquake

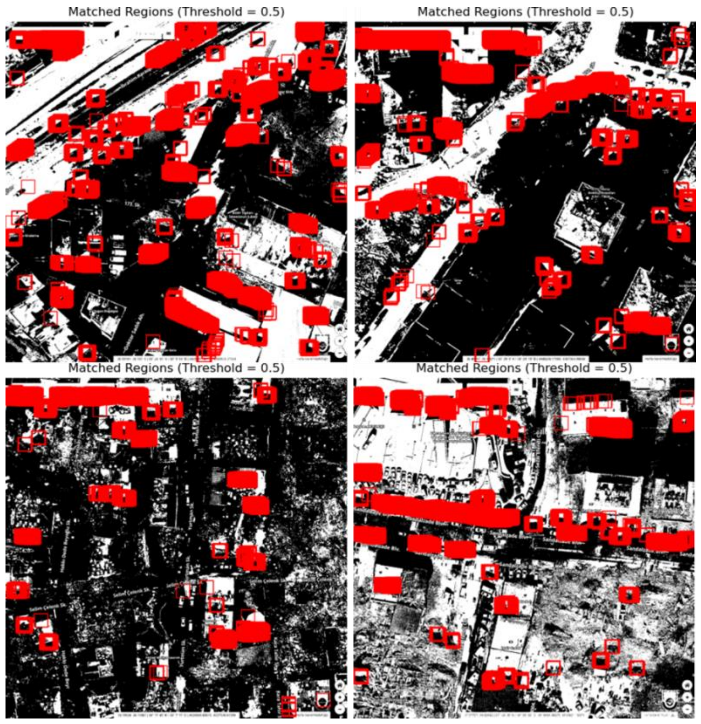

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.14** Matched Regions Detected by color based v Masks (Threshold 0.5)

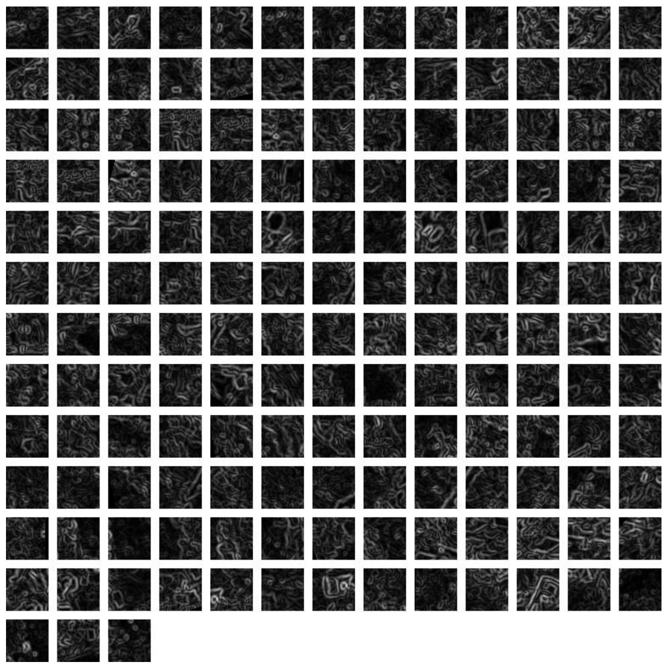


**Figure 3.15** Composite Mask of Debris Regions Identified by sobel Algorithm Post-Earthquake

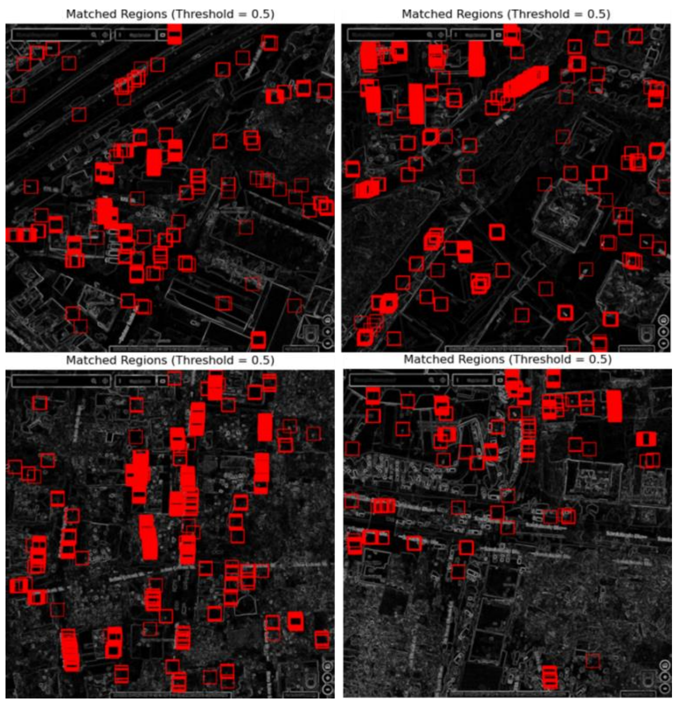

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.16** Matched Regions Detected by sobel Masks (Threshold 0.5)

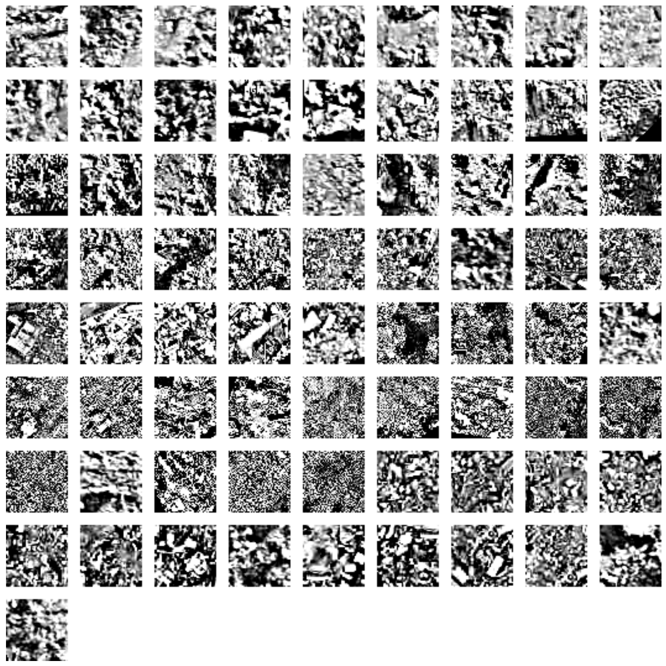


**Figure 3.17** Composite Mask of Debris Regions Identified by unsharp masking Algorithm Post-Earthquake

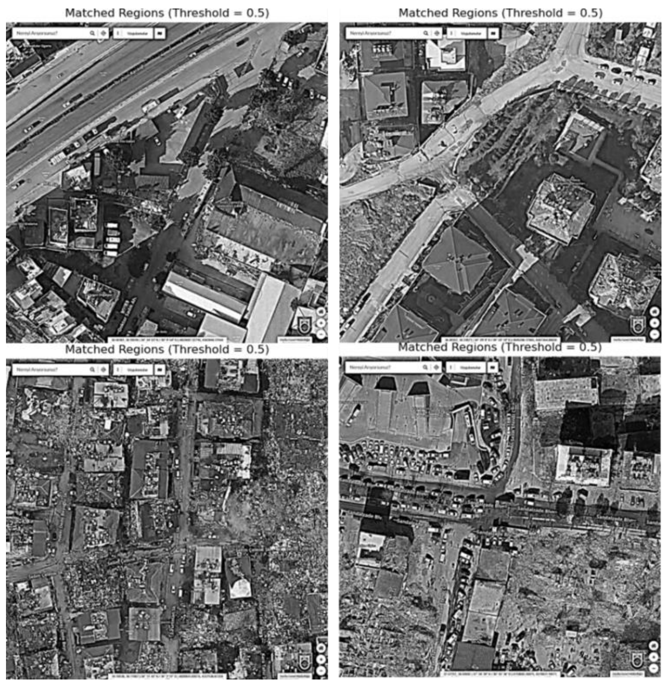

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.18** Matched Regions detected by unsharp masking Masks (Threshold 0.5)

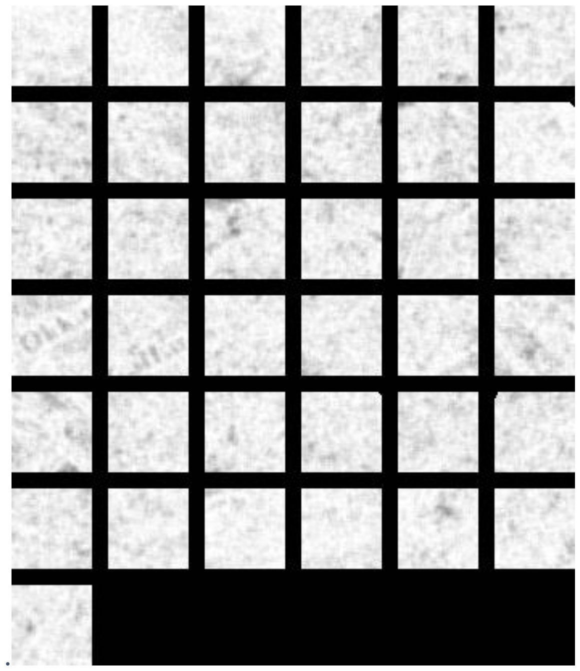


**Figure 3.19** Composite Mask of Debris Regions Identified by local entropy Algorithm Post-Earthquake

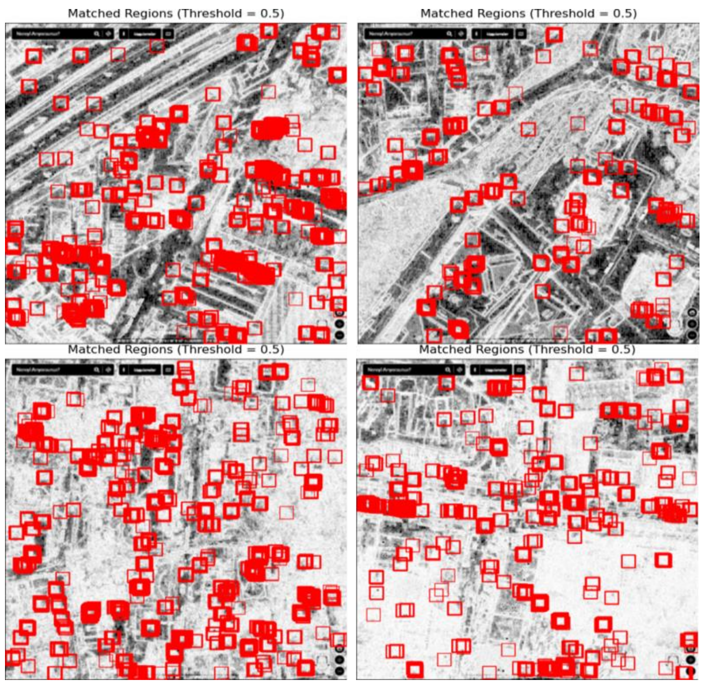

Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)
Matched Regions (Threshold = 0.5)

**Figure 3.20** Matched Regions Detected by color based local entropy Masks (Threshold 0.5)

As the primary objective of this study and thesis does not center on discerning differences through classical image processing, the focus remains on comparing the outcomes with those obtained through Deep learning methodologies. Consequently, the study did not endeavor to generate additional masks from diverse regions. It is acknowledged that expanding the dataset to include a greater number of images could potentially yield more satisfactory results. With an augmented dataset, it would be feasible to create an extensive array of masks, potentially exceeding 1000 masks for each algorithm. This increased dataset size could contribute to a more comprehensive and nuanced understanding of the algorithms' performance across various image contexts, enhancing the depth and reliability of the comparative analysis.

To create a mask within the graphical user interface (GUI), users can follow a simple process facilitated by the interface controls. First, the GUI displays the image of interest, providing users with a visual representation of the data. Users can then interact with the image by clicking on it to define the center of the region of interest (ROI). Upon clicking, a resizable rectangle appears, initially set to a default size. Users can adjust the size of the rectangle as needed, either increasing or decreasing its dimensions using dedicated controls. The rectangle is visually distinguished by a black outline, making it easy to identify within the image. Users can adjust the size of the rectangle by pressing the '+' or '-' keys to increase or decrease its dimensions, respectively. Once users are satisfied with the size and position of the rectangle, they can finalize the selection by pressing a designated button or key. This action generates a binary mask based on the dimensions and location of the rectangle, effectively isolating the selected region from the rest of the image. This mask can then be used for further analysis, processing, or visualization within the GUI or other applications. The entire mask creation process is demonstrated in Figure 3.5, providing users with a clear understanding of the steps involved.

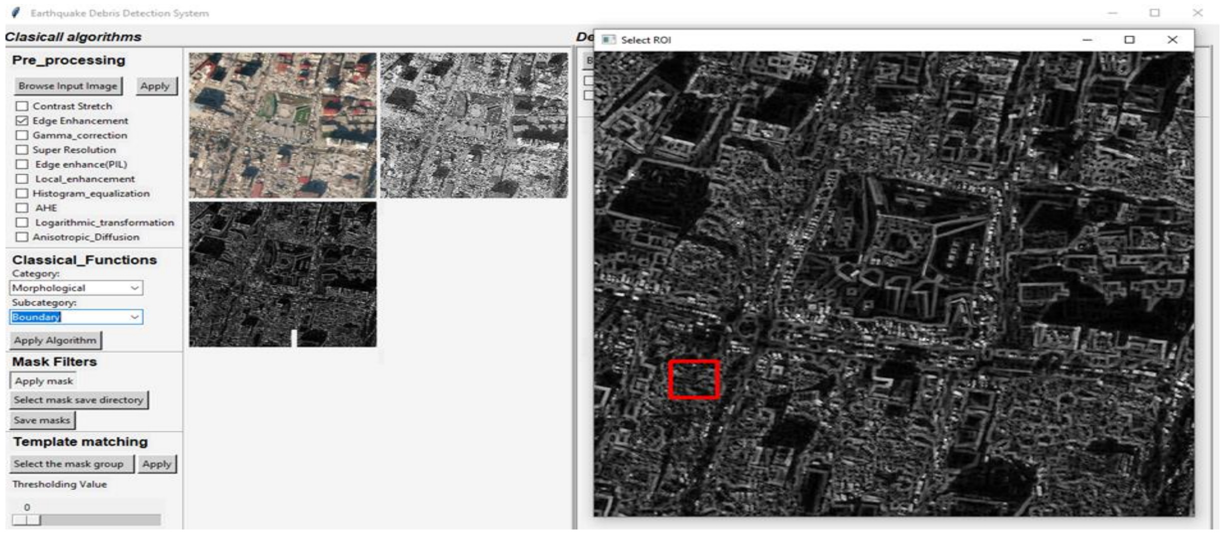
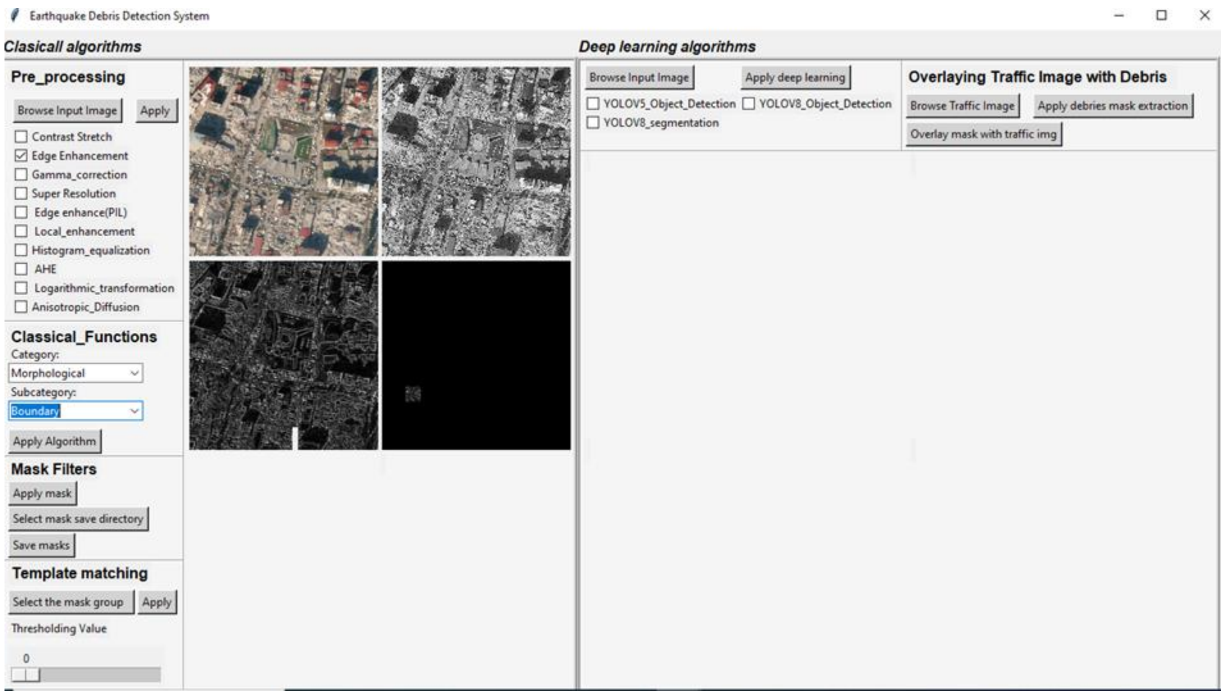


**Figure 3.21** Generating Masks through Interactive GUI

Once the desired area is selected, users can save the generated mask by specifying the address of the destination folder within the GUI interface. This feature facilitates seamless integration with template matching algorithms, enabling precise alignment and comparison of patterns within images. The created mask is shown in Figure 3.5, providing users with a visual reference of the steps involved.



☑ Edge Enhancement

Category: Morphological

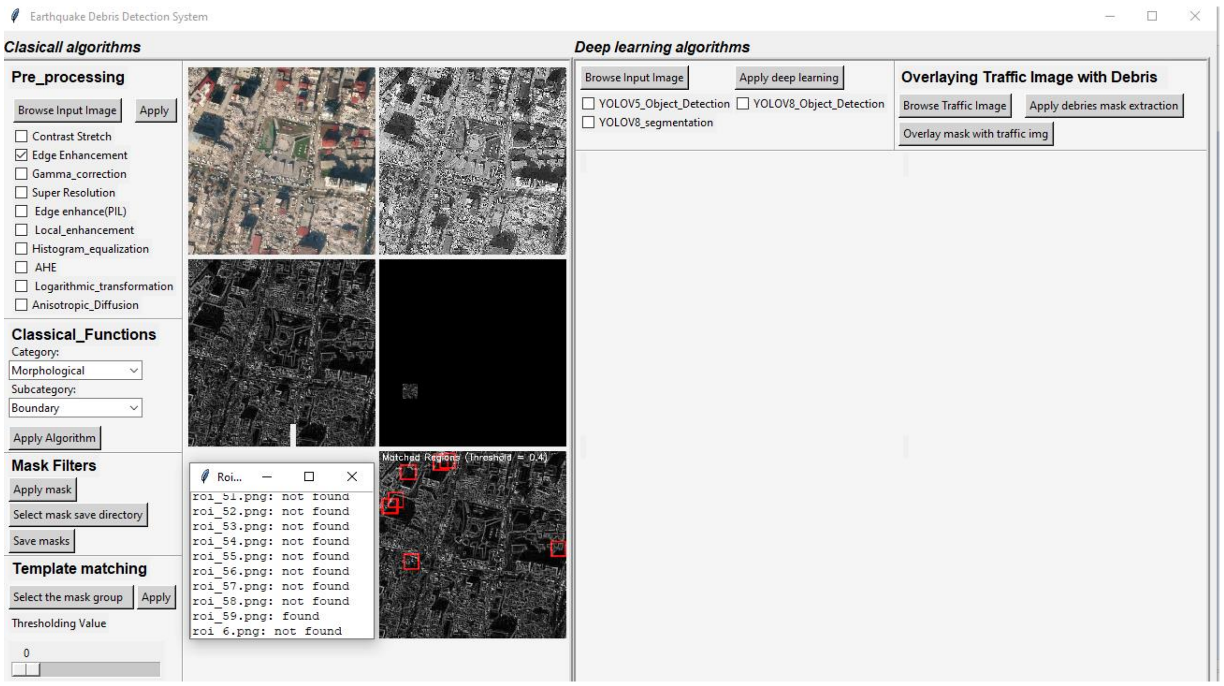
Subcategory: Boundary

**Figure 3.22** Save the mask for future use in template matching

In the graphical user interface (GUI) developed for this application, the utilization of correlation techniques initiates a process where a dedicated window dynamically presents a comprehensive overview of the identified masks within the analyzed image. Serving as an informative dashboard, this window effectively highlights the presence or absence of specific masks within the image, thus offering insights into whether a particular region of interest (ROI) is successfully detected. This functionality enables users to promptly discern which masks are effectively matched within the image and which ones are not.

Furthermore, the GUI interface offers valuable insights into the compatibility of the detected masks with potential debris. By leveraging this information, users can identify promising candidates that may aid in the precise localization and identification of debris within the analyzed imagery. This nuanced analysis and interpretation empower users to make informed decisions regarding the selection and utilization of masks for effective debris detection and analysis.

In the context of the template matching process illustrated in Figure 3.6, the GUI facilitates a visual representation of the correlation techniques in action, showcasing their impact on mask identification and utilization. Additionally, when a ROI mask is detected within an image through template matching, users have the opportunity to manually assess whether that mask represents debris. If confirmed, users can gather all masks representing debris as potential debris masks, optimizing the template matching process and enhancing its accuracy in identifying relevant debris within the imagery.



```
roi_51.png: not found
roi_52.png: not found
roi_53.png: not found
roi_54.png: not found
roi_55.png: not found
roi_56.png: not found
roi_57.png: not found
roi_58.png: not found
roi_59.png: found
roi 6.png: not found
```

**Figure 3.23** Template Matching Process Visualization

## 3.2 Deep learning Results

In this section, we present the outcomes of employing various Deep learning algorithms for object detection and Segmentation tasks. The results encompass a range of state-of-the-art algorithms, including YOLOv5 and YOLOv8 for object detection, Mask R-CNN with Detectron2, Roboflow Instance Segmentation, and YOLOv8 for object Segmentation. Each subsection discusses the specific algorithm's performance, accompanied by graphs illustrating key metrics such as precision, recall, loss functions, and mean average precision (mAP). Additionally, dataset prediction images are provided to visually depict the algorithm's effectiveness in detecting and segmenting objects of interest. Through comprehensive analysis and visualization, we evaluate the suitability and efficacy of each algorithm in addressing the target tasks. For training, Google Colab Premium was utilized, offering ample computational resources and an efficient execution environment.

### 3.2.1 Roboflow Data annotation and Augmentation Results

The dataset, comprising 510 satellite images initially sized at 1920x1080, underwent preprocessing for compatibility with YOLO. Resizing the images to 640x640, the augmentation process involved various techniques to enhance diversity and robustness. Augmentation methods included auto-orientation, resizing, flipping (horizontal and vertical), rotating (90° clockwise, counter-clockwise, and upside down), cropping (0-20% zoom), rotation (-15° to +15°), shearing (±15° horizontally and vertically), grayscale conversion (19% of images), hue adjustment (-23° to +23°), saturation modification (-30% to +30%), brightness alteration (-25% to +25%), exposure variation (-14% to +14%), and adding noise (up to 6% of pixels). Additionally, bounding boxes underwent similar transformations to maintain spatial correspondence. The dataset was divided into three sets: a training set (88% - 1077 images), a validation set (8% - 100 images), and a test set (4% - 51 images).

The four images in Figure 2.7 were selected from the annotated images available on the Roboflow website for utilization in Deep learning algorithms. Prior to this selection process, I uploaded all my satellite images to the Roboflow platform and meticulously annotated them. Subsequently, from this annotated dataset, I randomly chose four images to illustrate the quality and effectiveness of the annotation process on the Roboflow site. Leveraging Roboflow's annotation capabilities not only ensured the accurate labeling of objects and feature within the images but also facilitated the seamless integration of this annotated data into Deep learning workflows for model training and evaluation.

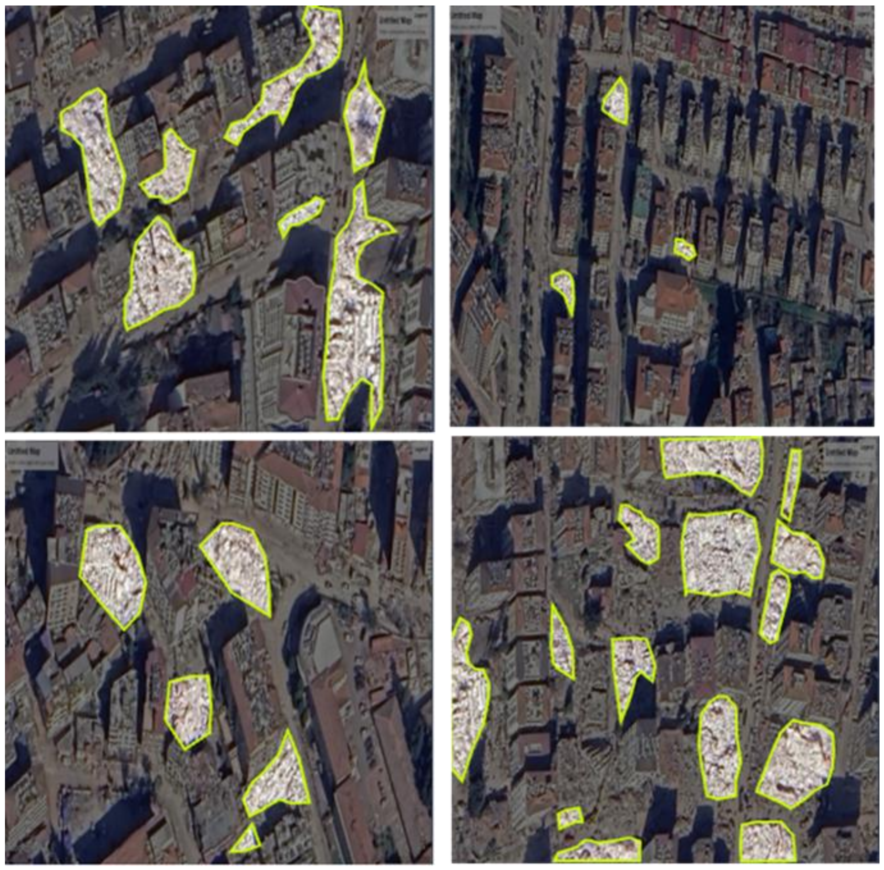


**Figure 2.24** Annotated Satellite Images from Roboflow

### 3.2.2 Object Detection with YOLO Results

Visualizing the outcomes through various graphs, depicting metrics such as precision, recall, and accuracy over different epochs or evaluation criteria, offers insightful information about the performance and effectiveness of the YOLO model in detecting objects across various scenarios and datasets. By analyzing these graphs, we gain the ability to assess the model's strengths, weaknesses, and areas for improvement, thereby advancing the field of computer vision and object detection. In the following sections, I will provide a detailed explanation of each graph, elucidating its significance and implications for the YOLO model.

a) Train/box_loss: This graph shows the loss associated with bounding box predictions during the training process. It indicates how well the model is able to localize objects within the images. A decreasing trend in this loss suggests that the model is improving in accurately predicting the bounding boxes of objects.

b) Train/obj_loss: This graph represents the loss related to objectness prediction during training. It reflects the model's ability to differentiate between object and background regions in the images. Decreasing values of this loss indicate that the model is becoming more proficient at identifying relevant objects.

c) Train/cls_loss: The train/cls_loss graph displays the loss attributed to class prediction during training. It indicates how well the model is performing in classifying detected objects into different categories or classes. Lower values of this loss signify better classification accuracy.

d) Metrics/precision: This graph illustrates the precision of the model's predictions. precision measures the accuracy of positive predictions made by the model, indicating the proportion of true positive predictions among all positive predictions. Higher precision values indicate fewer false positives.

e) Metrics/recall: The metrics/recall graph shows the recall of the model's predictions. Recall measures the ability of the model to correctly identify all relevant instances, indicating the proportion of true positives predicted among all actual positives. Higher recall values suggest fewer false negatives.

f) Val/box_loss, Val/obj_loss, Val/cls_loss: These graphs are comparable to their respective training counterparts but represent the loss values on the validation dataset. They provide insight into the model's performance on unseen data during validation.

g) *Metrics/mAP_0.5:* This graph shows the (mAP) at an (IoU) threshold of 0.5. It evaluates the overall performance of the model in object detection, considering both precision and recall across different object categories.

h) Metrics/mAP_0.5:0.95: This graph presents the mAP across a range of IoU thresholds from (0.5-0.95) It provides a comprehensive evaluation of the model's performance, considering varying levels of overlap between predicted and ground-truth bounding boxes.

i) F1 Confidence Curve: This curve shows the connection between the F1 score and confidence thresholds used for object detection or classification tasks. It helps in determining the optimal confidence threshold that balances precision and recall to achieve the highest F1 score, thereby providing insight into the model's performance across different confidence levels.

j) Precision-Confidence Curve:The precision-confidence curve showcases how precision varies with different confidence thresholds. It demonstrates the trade-off between precision and confidence, helping to identify the threshold that achieves the desired precision level for decision-making in object detection or classification tasks.

k) Precision-Recall Curve: The trade-off between precision and recall is shown by the graph. at different confidence thresholds. It provides a comprehensive view of the model's performance, allowing analysts to choose the appropriate threshold based on the specific requirements of their application. A higher area under the precision-recall curve indicates better model performance in balancing precision and recall.

l) Recall:Recall, also known as sensitivity, is a statistic used to assess a model's capacity. to correctly identify all relevant instances, including true positives, out of all actual positive instances in the dataset. It is calculated as the ratio of true positives to the sum of true positives and false negatives. A higher recall indicates that the model can effectively capture a larger proportion of positive instances, minimizing false negatives .

m) Precision: Precision gauges how well the model predicts favorable occurrences. among all predicted positive instances. It is calculated as the ratio of true positives to the sum of true positives and false positives. A higher precision indicates that the model makes fewer false positive predictions, providing more confidence in the correctness of its positive predictions.

n) F1 Confidence:F1 confidence is a metric that combines precision and recall into a single score, providing a balance between the two. It is the harmonic mean of precision and recall and is calculated as 2× precision×recall/ precision+recall . F1 confidence is useful when there is an uneven class distribution or when both precision and recall are important metrics for evaluating model performance[47] .

o) Confusion matrix:A confusion matrix is a evaluation of performance tool for machine learning classification problems where the output can have two or more classes. It is a table that provides a detailed breakdown of of accurate and inaccurate predictions made by a classification model. The matrix consists of four different combinations of predicted and actual class values: true positives (correctly predicted positive instances), false positives (incorrectly predicted positive instances), true negatives (correctly predicted negative instances), and false negatives (incorrectly predicted negative instances). The examples in a predicted class are represented by each column of the matrix, whereas the occurrences in an actual class are represented by each row. The confusion matrix is a useful tool for evaluating the performance of a classification model, providing insights into its strengths and weaknesses across different classes.

p) *Labels correlations:*In the context of object detection and classification, labels correlations refer to the relationships or dependencies between different categories or classes of objects within a dataset. These correlations indicate how often certain labels co-occur or share similar visual characteristics. Understanding label correlations is crucial for developing accurate and robust models, as it helps in predicting the presence of one object based on the presence of another related object. For example, in an image containing a person, there might be a high correlation with other objects such as a backpack or a bicycle. By leveraging label correlations, object detection models can improve their accuracy and performance, leading to more reliable results in real-world applications.

#### 3.2.2.1. Object Detection with YOLOv5 Results

In the final epoch (epoch 59) of training the YOLOv5m model, several key performance metrics were evaluated. The train/box_loss and train/obj_loss achieved low values of 0.029389 and 0.038873, respectively, indicating effective optimization of bounding box predictions and abjectness scores. Notably, the model achieved a precision of 56.45% and a recall of 63.03%, demonstrating its ability to accurately detect and classify objects within images. The (mAP) at an IoU threshold of 0.5 (mAP_0.5) was measured at 57.105%, showcasing the model's effectiveness in detecting objects with varying levels of overlap. Furthermore, the mAP_0.5:0.95 values, which consider a broader range of IoU thresholds, reached 28.262%, suggesting consistent performance across different IoU thresholds. Validation losses (val/box_loss and val/obj_loss) remained low at 0.046376 and 0.055355, respectively, indicating the model's ability to generalize well to unseen data. The learning rate parameters (x/lr0, x/lr1, x/lr2) remained consistent at 0.00043 throughout training, suggesting stable learning dynamics. Overall, the results of epoch 59 demonstrate the YOLOv5m model's robustness and effectiveness in object detection tasks, with promising performance across various evaluation metrics.

The Figure 3.5 containing nine test data predictions offers valuable insights into the efficacy of YOLOv5m in detecting debris post-earthquake.YOLOv5m successfully identifies regions of interest through object detection boxes, showcasing precision scores ranging from 0.25 to 0.87. However, it's notable that precision levels fluctuate across different debris detections due to the diverse textures and appearances of the debris compared to the surrounding environment.

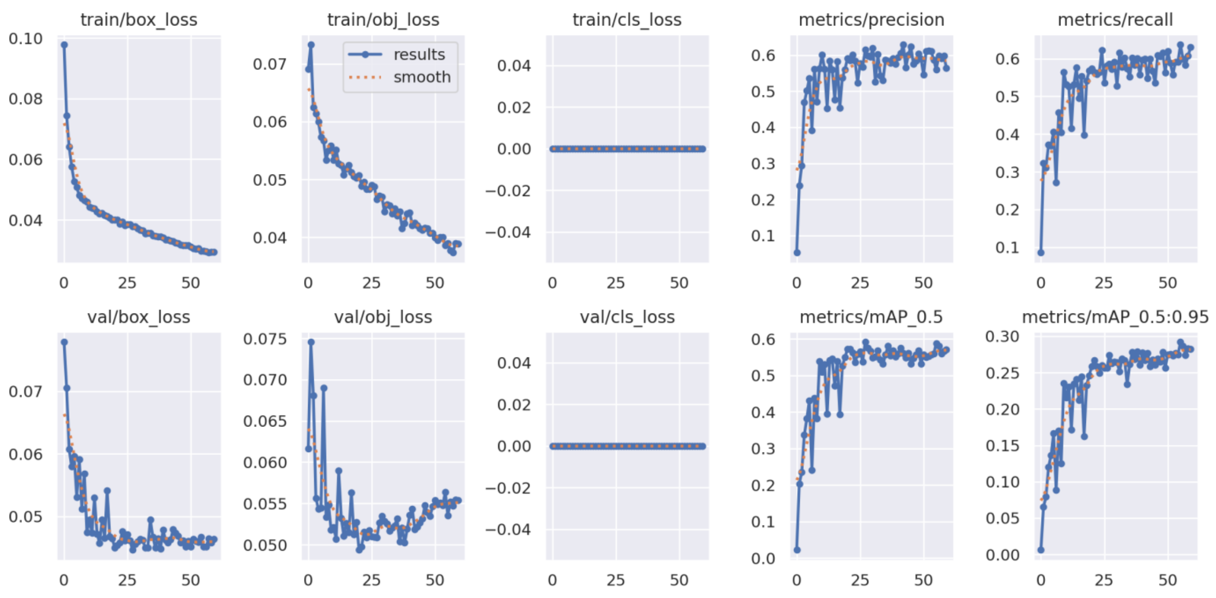


**Figure 3.25** YOLOv5m object detection training Insights and Performance Metrics

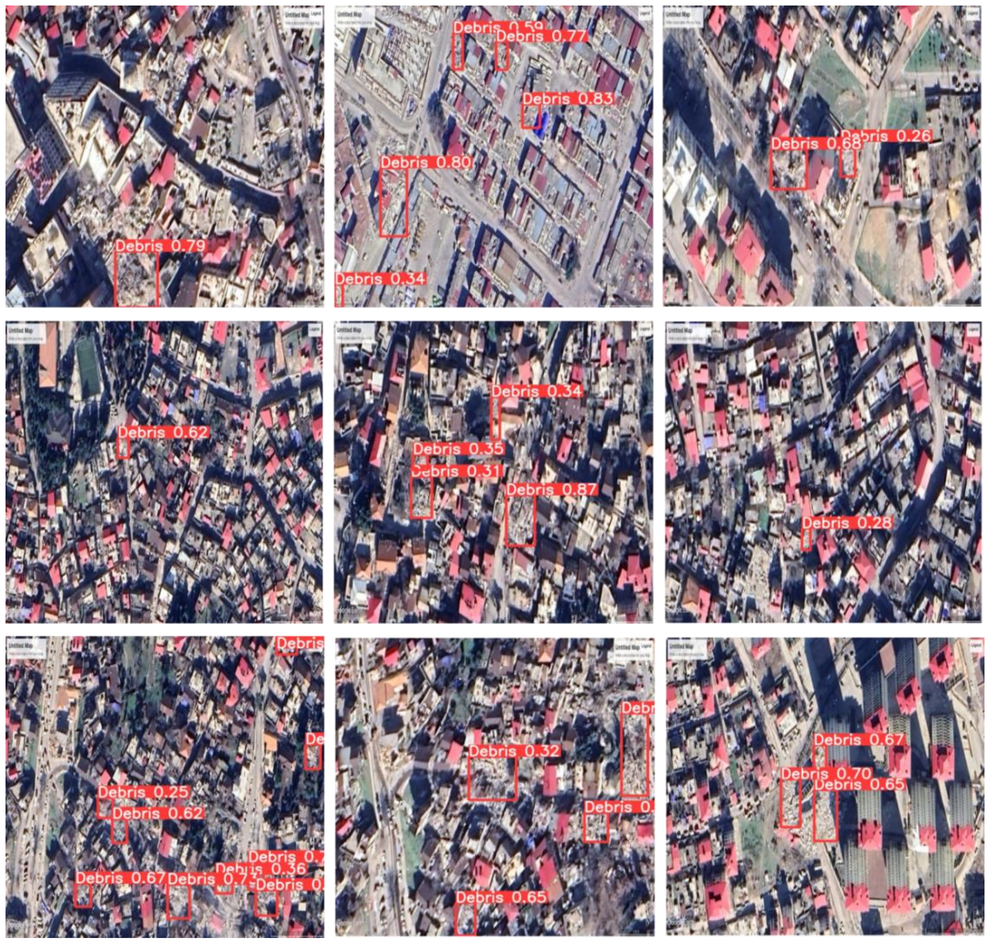


**Figure 3.26** Test data prediction with YOLOv5m object detection

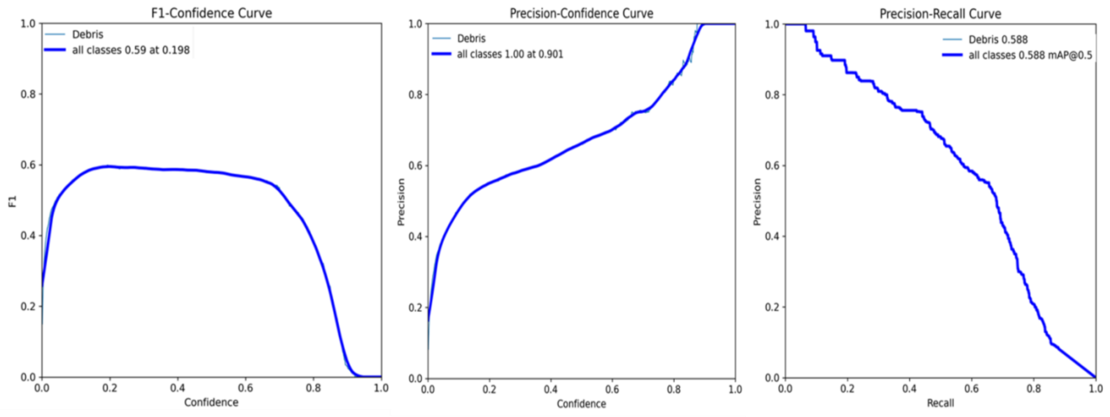


**Figure 3.27** Left to right: F1 Confidence, precision-Confidence, and precision-Recall curves results with YOLOv5m object detection

In the YOLOv5m object detection confusion matrix for debris detection, the confidence scores provide insights into the model's performance in distinguishing between debris and background classes. A confidence score of 0.65 is associated with the debris class, indicating a high level of certainty when the model identifies an object as debris. This suggests that the model is correct about 65% of the time when predicting debris. Conversely, a confidence score of 0.35 is assigned to the background class, implying a lower level of certainty compared to debris predictions. When the model predicts an area as background, it is correct about 35% of the time. These confidence scores reflect the model's ability to accurately differentiate between debris and background, with higher scores indicating greater confidence in the predictions.

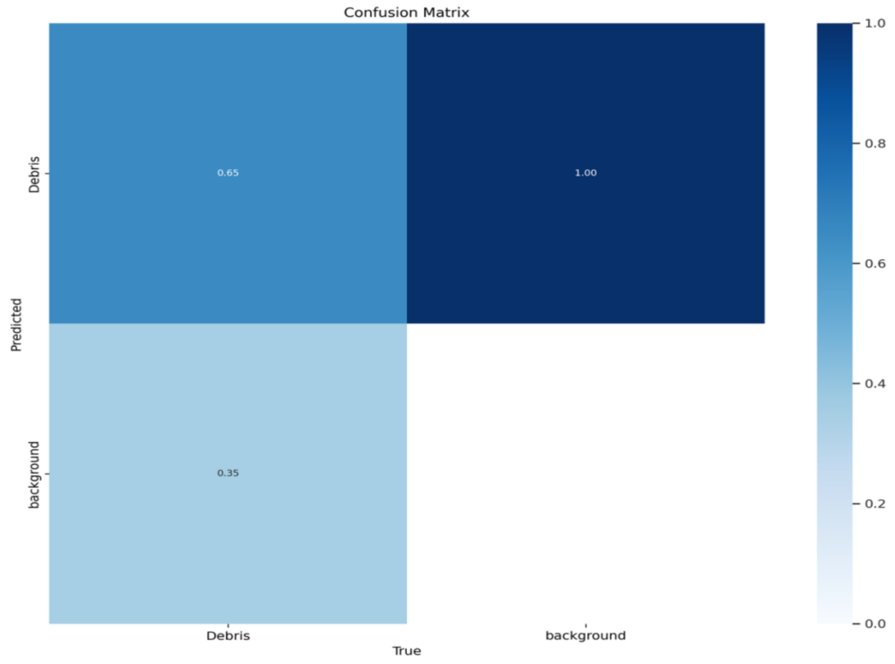


**Figure 3.28** YOLOv5m object detection confusion matrix

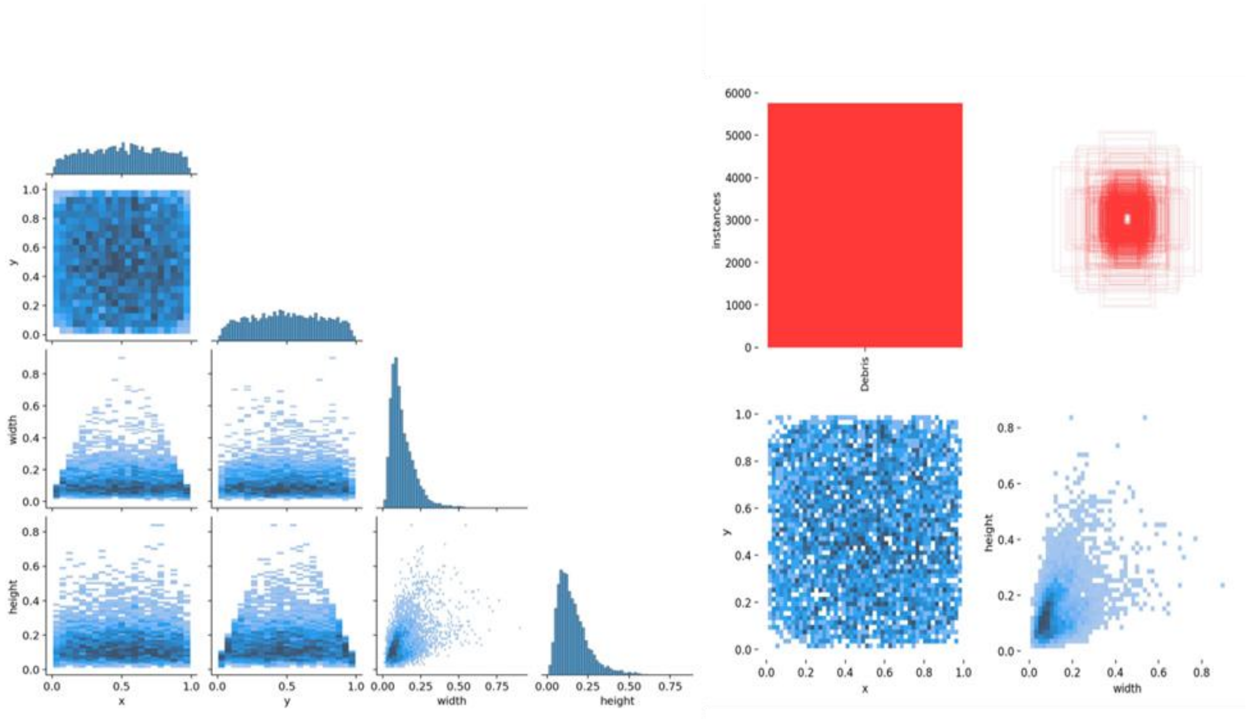
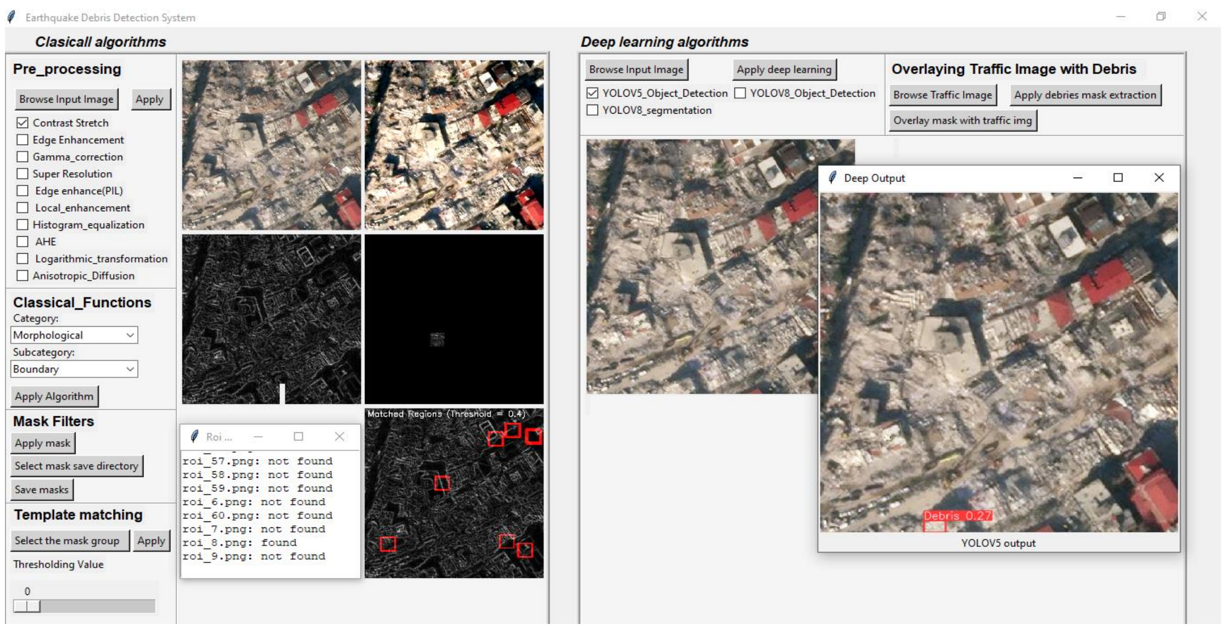


**Figure 3.29** YOLOv5m object detection labels correlations



☑ Contrast Stretch

Category: Morphological

Subcategory: Boundary

Thresholding Value: 0

☑ YOLOV5_Object_Detection

```
roi_57.png: not found
roi_58.png: not found
roi_59.png: not found
roi_6.png: not found
roi_60.png: not found
roi_7.png: not found
roi_8.png: found
roi_9.png: not found
```

**Figure 3.30** GUI Display of YOLOv5 Object Detection Result

### 3.2.2.2. Object Detection with YOLOv8 Results

In the final epoch of training the YOLOv8 model, the performance metrics were evaluated. The Train/box_loss and Train/obj_loss reached 1.6325 and 1.2281, respectively, indicating relatively higher losses compared to YOLOv5m, suggesting potential challenges in bounding box predictions and objectness scores optimization. The Metrics/precision achieved 58.856%, while Metrics/recall attained 47.541%, demonstrating moderate performance in object detection and classification tasks. The Metrics/mAP_0.5 and Metrics/mAP_0.5:0.95 reached 51.988% and 23.239%, respectively, suggesting lower performance compared to YOLOv5m, particularly in detecting objects with higher IoU thresholds. Validation losses, Val/box_loss and Val/dfl_loss, remained at 0, indicating potential overfitting or inadequate generalization to unseen data. Learning rates (LR/pg0, LR/pg1, LR/pg2) remained constant at 0.00043559 throughout training, implying stable learning dynamics. Additionally, it's worth noting that following these results, there are 9 test image prediction results of YOLOv8, providing further insight into the model's performance.

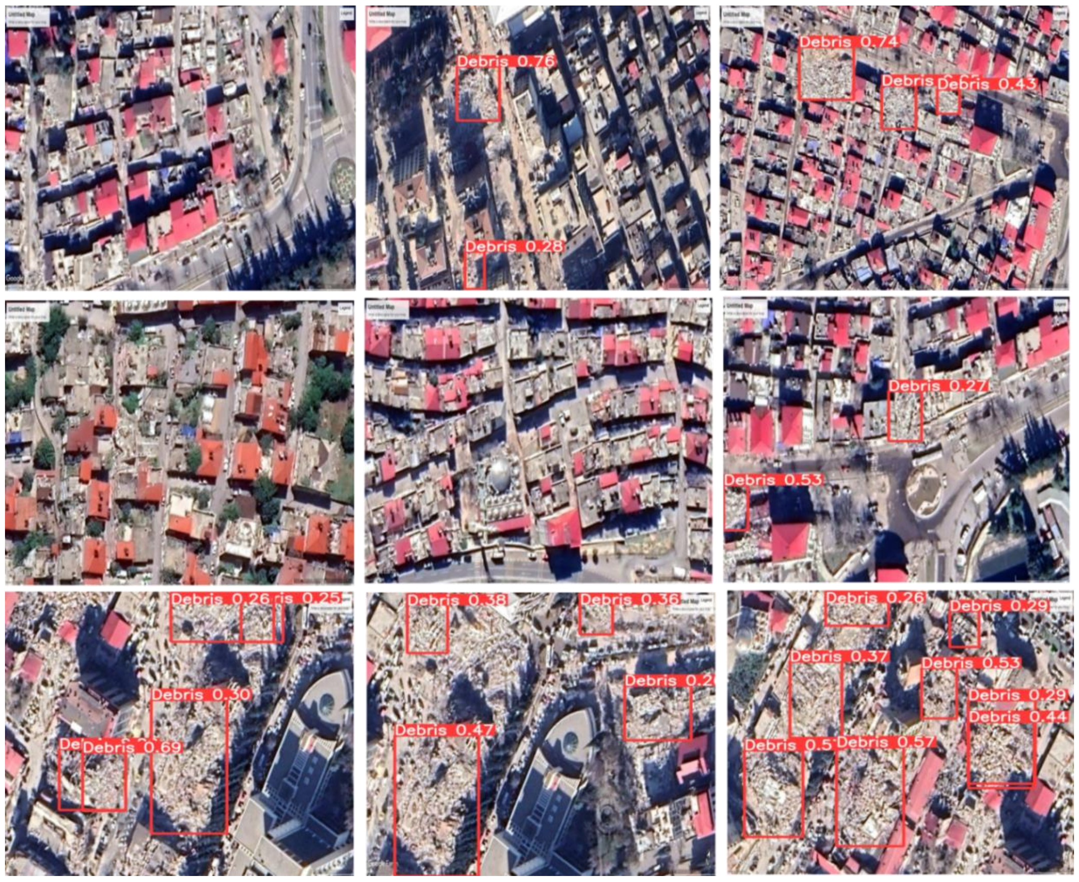


Figure 3.31 Test data prediction with YOLOv8 objects detection

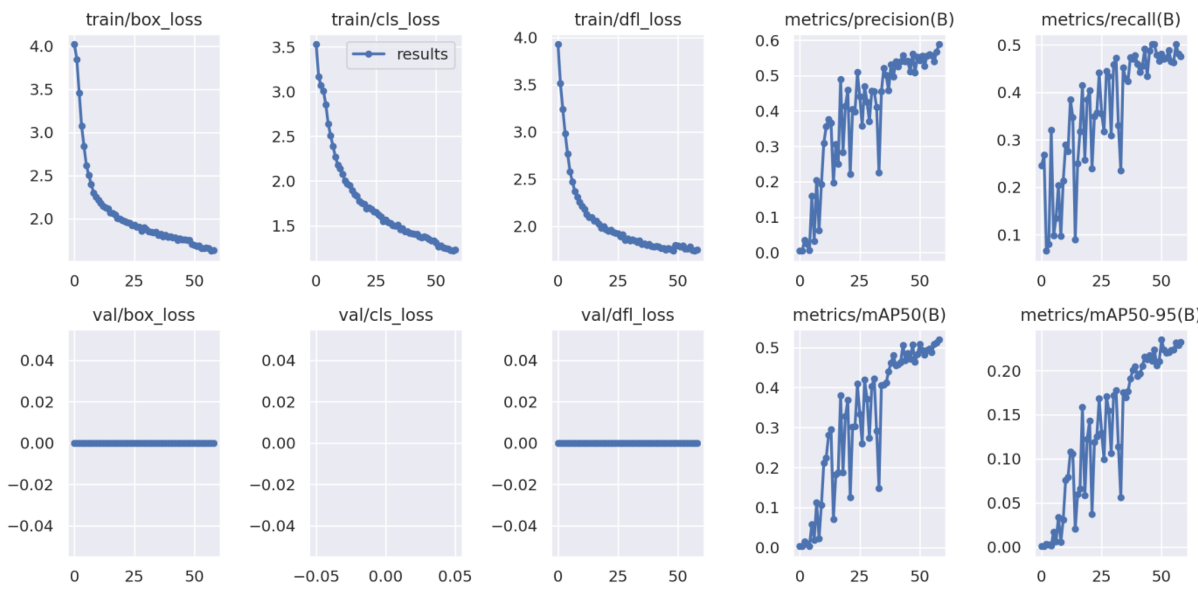


**Figure 3.32** YOLOv8 object detection training Insights and Performance Metrics

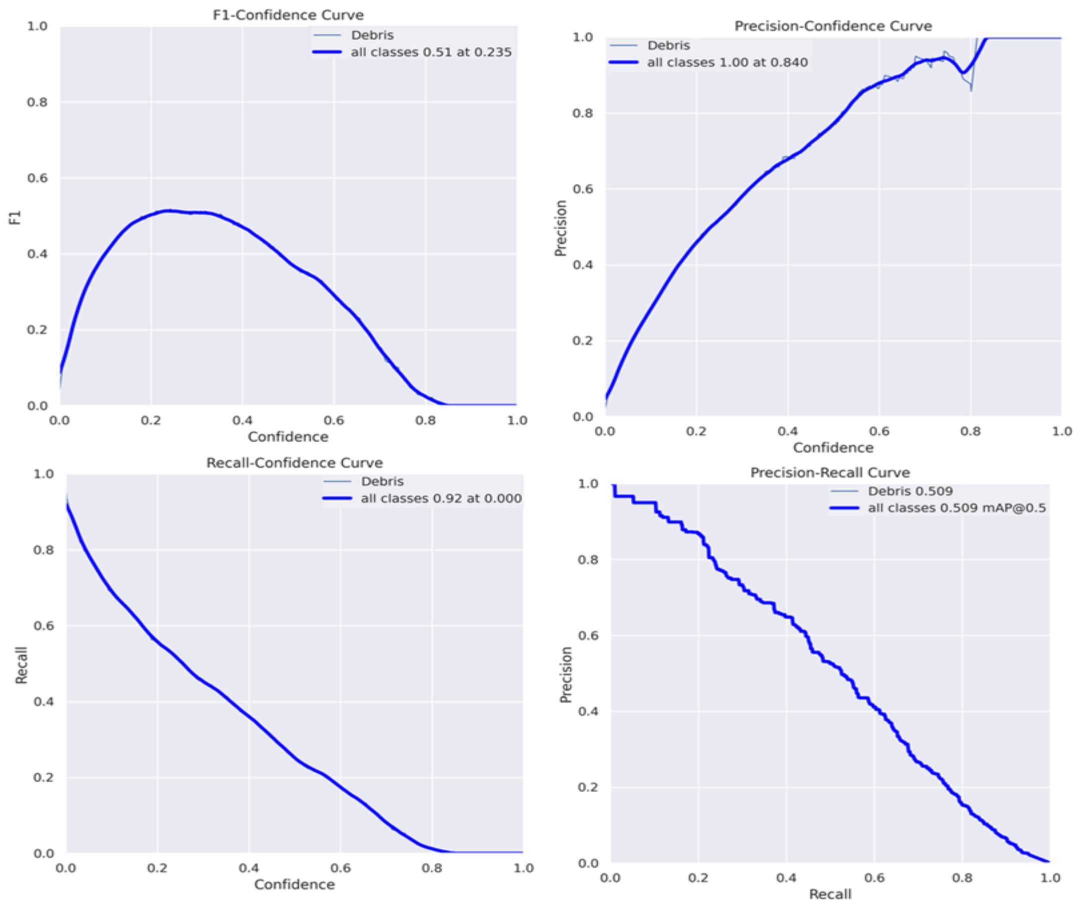


**Figure 3.33** YOLOv8 object detection Performance Evaluation Curves

In the YOLOv8m object detection confusion matrix, the value of 0.55 attributed to Debris signifies a moderate level of confidence in the model's prediction when identifying an object as debris. This score indicates that the model correctly identifies debris approximately 55% of the time, reflecting a reasonable degree of accuracy in its predictions. Conversely, the score of 0.45 assigned to Background suggests a lower level of confidence when the model predicts an area as background. In such instances, the model is accurate about 45% of the time. When comparing these results with the YOLOv5m model, which achieved a higher confidence score of 0.65 for debris and a lower score of 0.35 for background, it appears that the YOLOv5m model demonstrates greater confidence in its predictions, particularly for identifying debris. The higher confidence score for debris indicates a higher accuracy rate compared to the YOLOv8m model, while the lower confidence score for background implies a similar level of uncertainty in both models' background predictions.

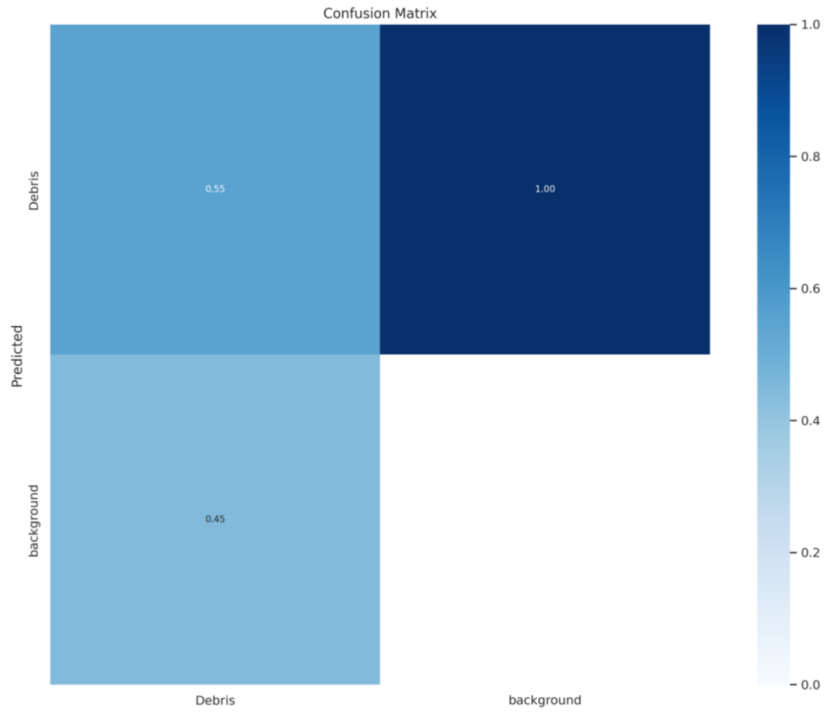
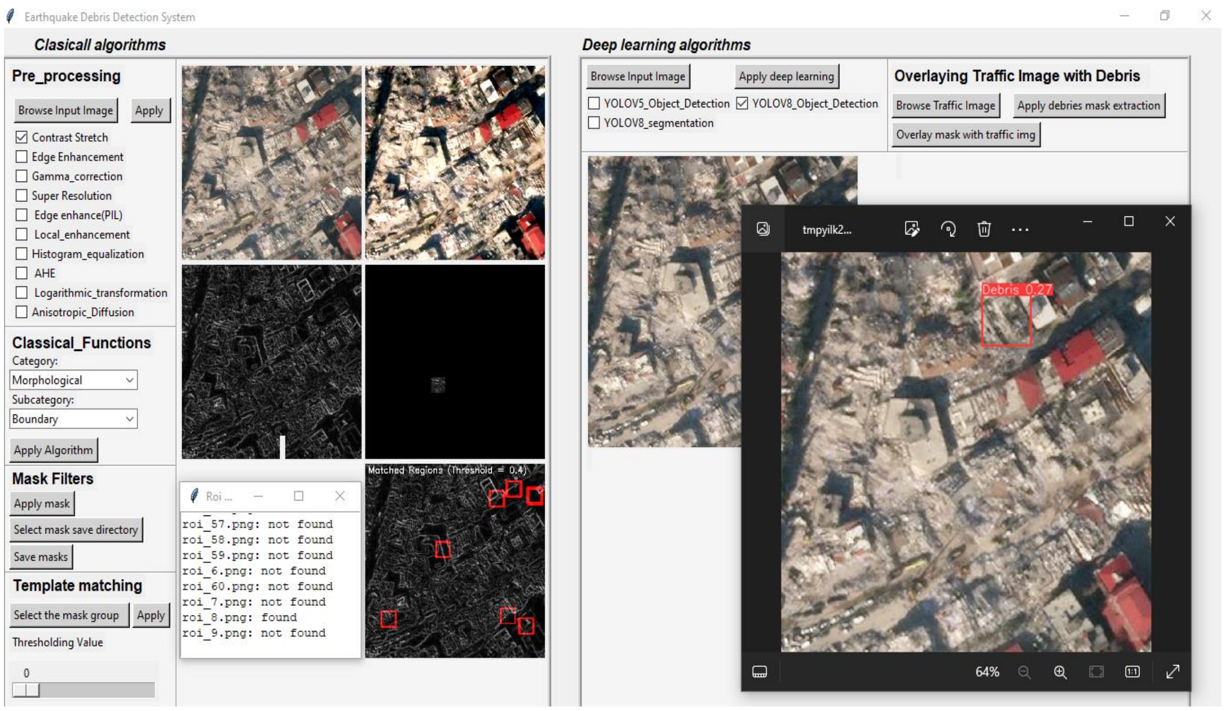


**Figure 3.34** YOLOv8 object detection confusion matrix



```
roi_57.png: not found
roi_58.png: not found
roi_59.png: not found
roi_6.png: not found
roi_60.png: not found
roi_7.png: not found
roi_8.png: found
roi_9.png: not found
```

**Figure 3.35** GUI Display of YOLOv8 Object Detection Result

#### 3.2.2.3 YOLOv8 Object Segmentation Results

In the final epoch (epoch 59) of training the YOLOv8 model for Segmentation tasks, several key performance metrics were assessed. The train/box_loss was recorded at 1.8554, indicating the optimization of bounding box predictions, while the train/seg_loss was 3.5162, representing the Segmentation loss during training. Additionally, the train/cls_loss was noted at 11.5774, reflecting the classification loss incurred during training. The model exhibited a precision of 51.313% and a recall of 45.173%, indicating its ability to accurately classify and capture segmented objects. The average precision at mean (mAP) threshold of 0.5 for IoU (mAP_0.5) was evaluated at 45.756%, while the mAP_0.5:0.95 was measured at 19.717%, suggesting moderate performance across different IoU thresholds. Furthermore, the Val/box_loss and Val/seg_loss maintained relatively high values at 1.9369 and 3.1221, respectively, indicating areas for potential improvement in validation losses. The learning rate parameters (lr/pg0, lr/pg1, lr/pg2) remained consistent at 8.71E-05 throughout training, suggesting stable learning dynamics. Overall, while the YOLOv8 model showed promising performance in certain aspects such as precision and recall, there is room for enhancement in reducing validation losses and improving mAP scores to enhance its effectiveness in Segmentation tasks. Additionally, the subsequent evaluation of four test image predictions will provide further insights into the model's generalization capabilities and practical applicability.

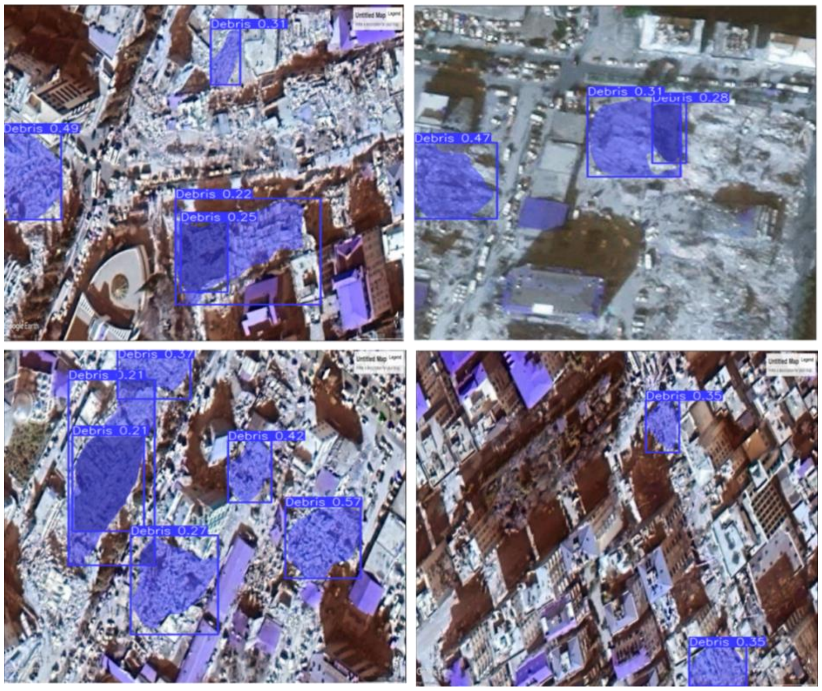


**Figure 3.36** YOLOv8 segmentation Test data prediction after 60 epochs

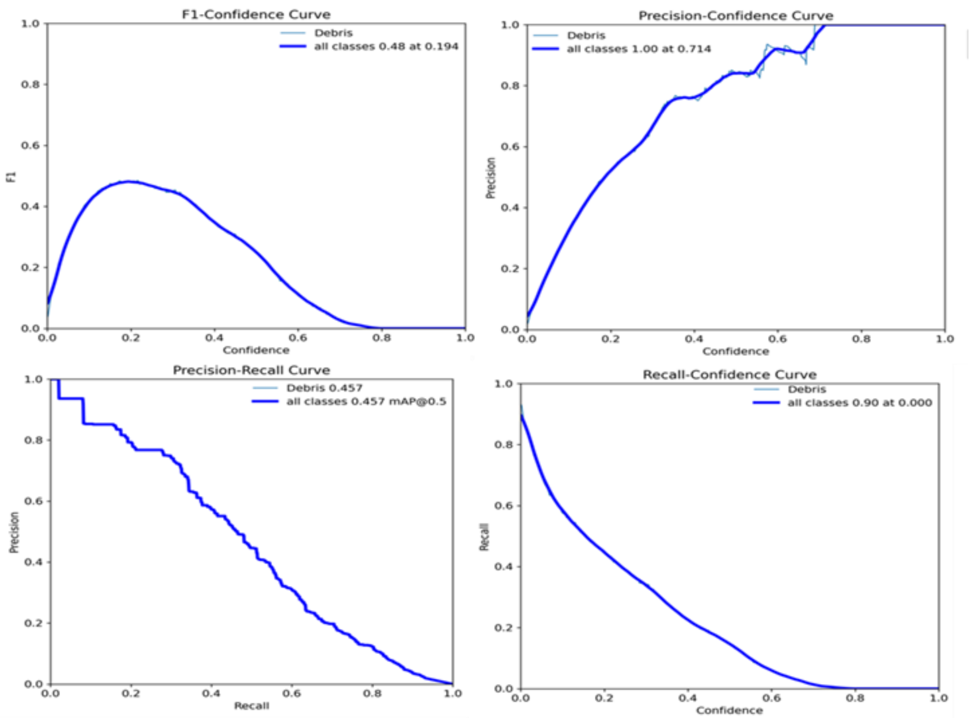


**Figure 3.37** YOLOv8 segmentation after 60 epochs Performance Evaluation Curves

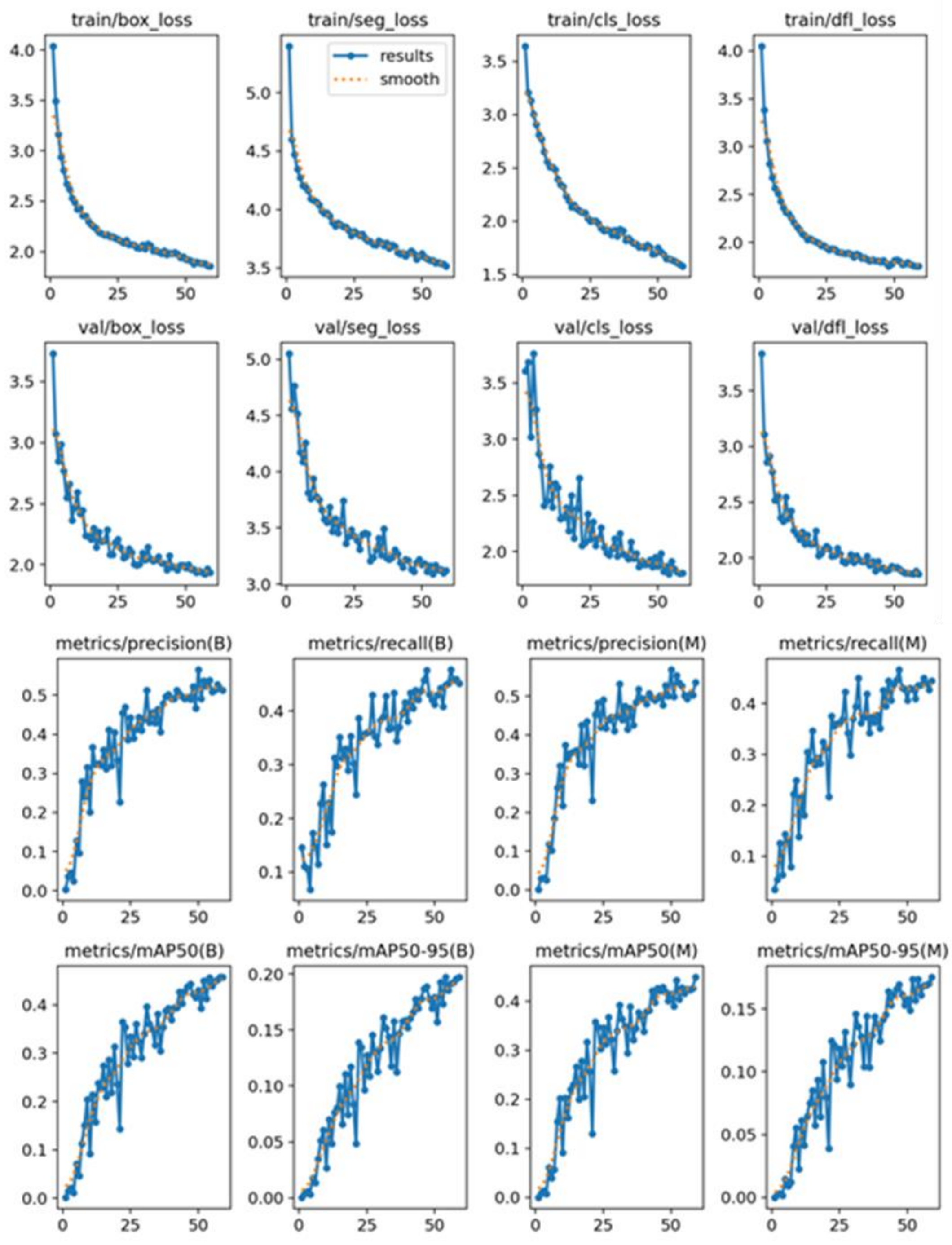


**Figure 3.38 YOLOv8** segmentation after 60 epochs training Insights and Performance Metrics

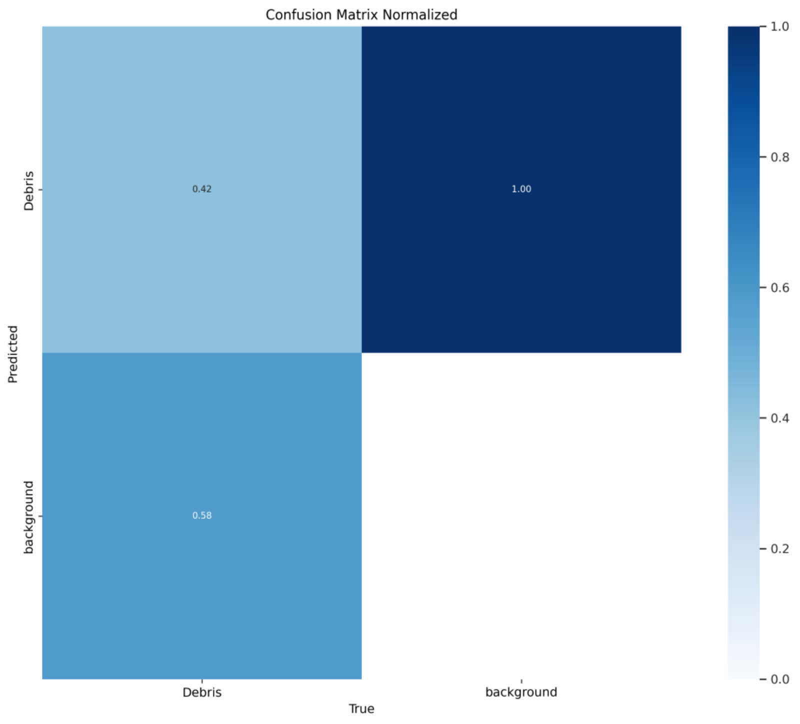


**Figure 3.39** YOLOv8 segmentation after 60 epoch's confusion matrix

In the 200th epoch of training the YOLOv8 Segmentation model, significant improvements were observed across various key metrics compared to the results from epoch 59. The train/box_loss decreased to 1.4879, indicating enhanced optimization of bounding box predictions. Similarly, the Segmentation loss (train/seg_loss) decreased to 2.9593, reflecting improved Segmentation accuracy. Notably, the model achieved a higher precision of 61.913% and recall of 48.998%, demonstrating enhanced object detection capabilities. Moreover when the IoU threshold is reached, the mean average precision (mAP) of 0.5 (mAP_0.5) increased to 53.818%, indicating improved object localization accuracy. The mAP_0.5:0.95 also saw a notable improvement, reaching 25.358%, suggesting enhanced performance across a broader range of IoU thresholds. In terms of validation losses (val/box_loss, val/seg_loss, val/cls_loss, val/dfl_loss), reductions were observed, indicating better generalization to unseen data. The learning rate parameters remained consistent throughout training. Overall, the results of the 200th epoch demonstrate the YOLOv8 Segmentation model's continued learning and improved performance over extended training periods. Comparatively, these advancements highlight the model's ability to refine its object detection and Segmentation capabilities over prolonged training durations.

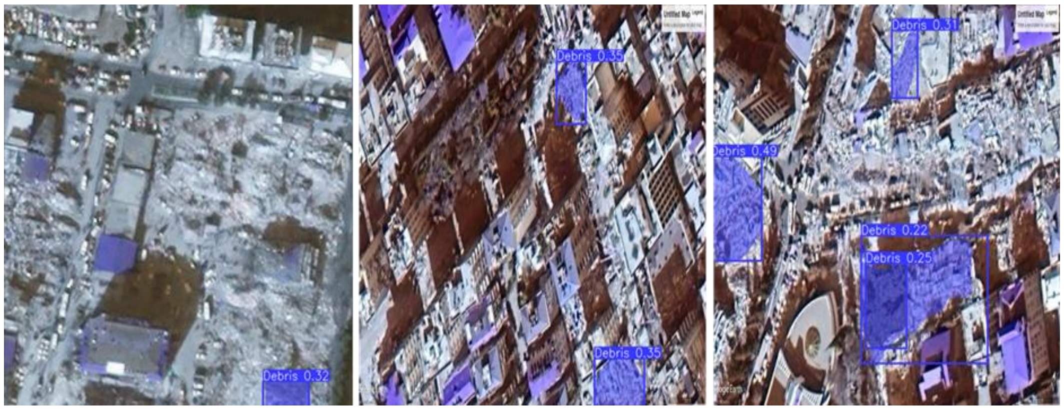


**Figure 3.40** YOLOv8 segmentation Test data prediction after 200 epochs

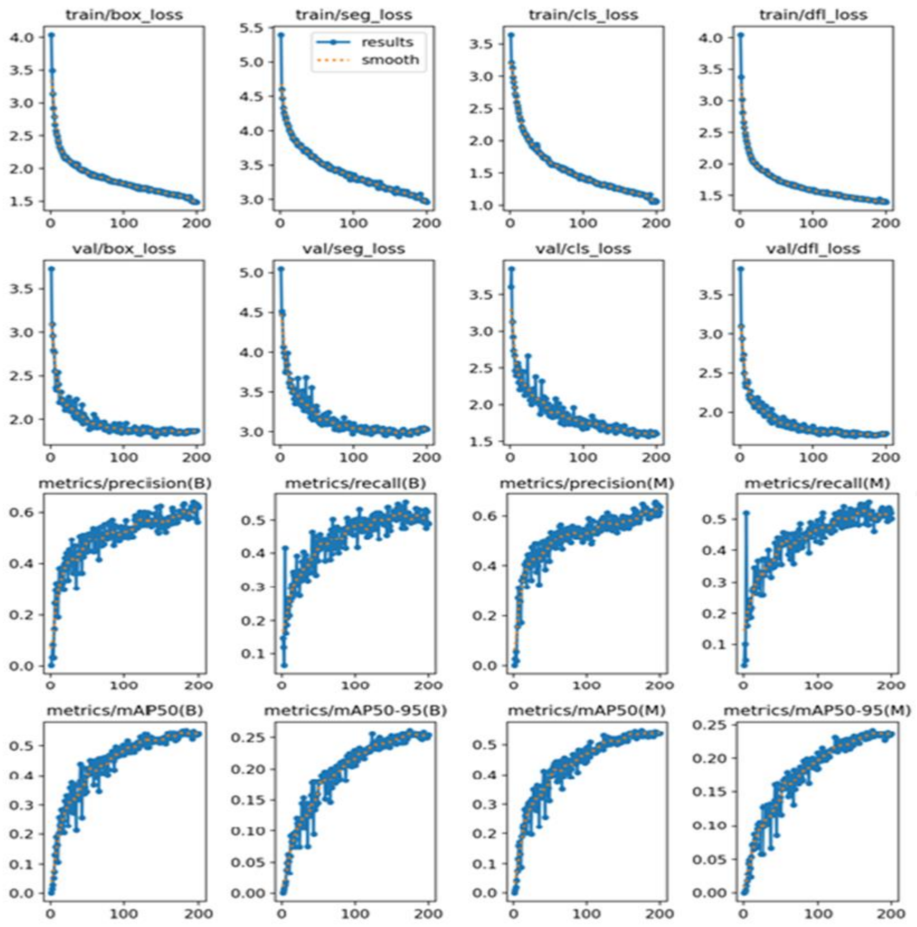


**Figure 3.41** YOLOv8 segmentation after 200 epochs training Insights and Performance Metrics

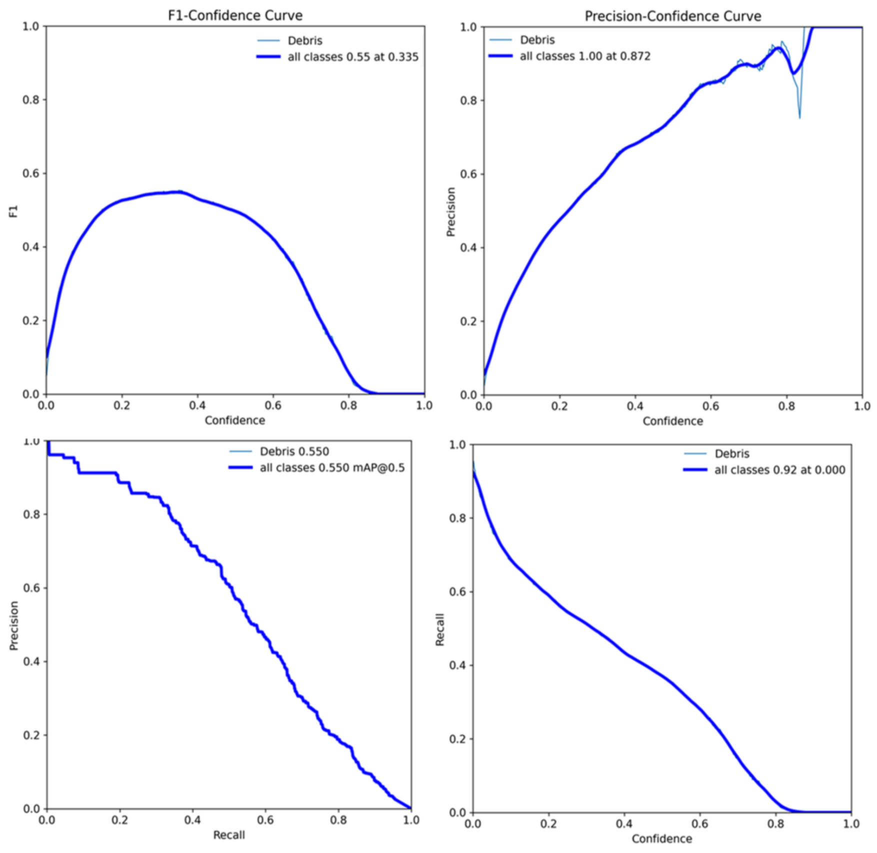


**Figure 3.42** YOLOv8 segmentation after 200 epochs Performance Evaluation Curves

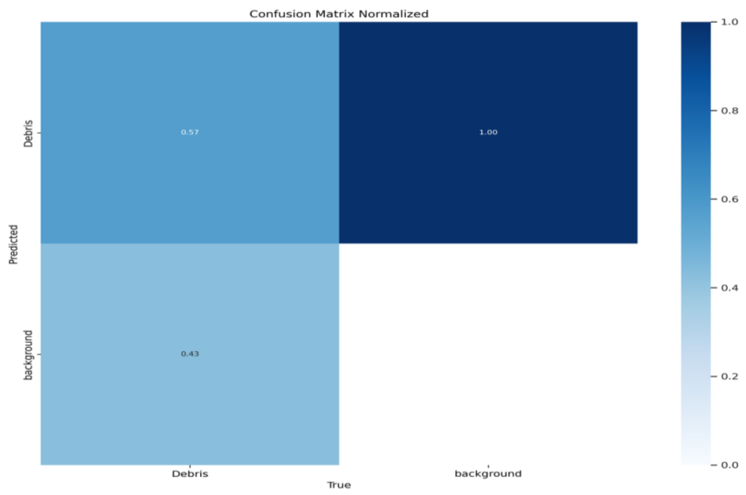
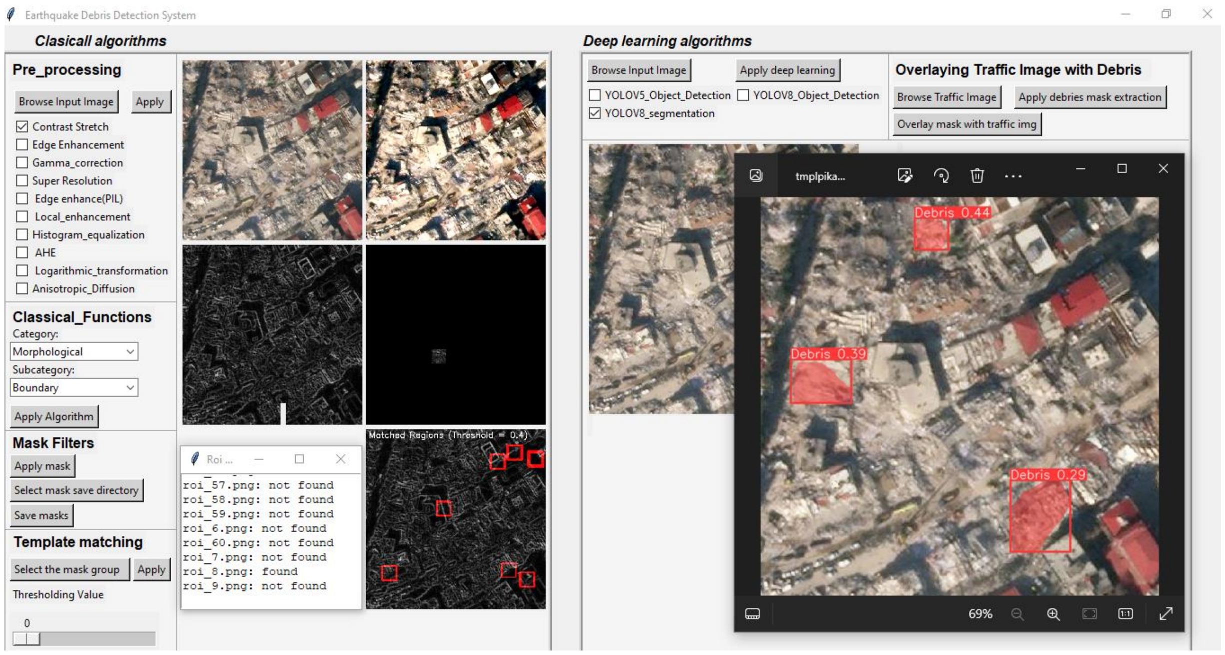


**Figure 3.43** YOLOv8 segmentation after 200 epochs confusion matrix



```
roi_57.png: not found
roi_58.png: not found
roi_59.png: not found
roi_6.png: not found
roi_60.png: not found
roi_7.png: not found
roi_8.png: found
roi_9.png: not found
```

**Figure 3.44** GUI Display of YOLOv8 segmentation Result

### 3.2.3 Mask R-CNN with Detectron2 Segmentation Results

The Mask R-CNN model underwent evaluation on the test dataset, with training epochs set at 4 and 1500. Performance metrics were computed using the COCO evaluation method.

The model achieved an Average precision (AP) of 6.61% across IoU thresholds ranging from (0.50-0.95), with a notable precision of 22.04% at IoU=0.50. At a higher IoU threshold of 0.75, the precision dropped to 2.78%. The Average Recall (AR) at IoU=0.50:0.95 was calculated to be 28.9%. Notably, the model demonstrated an AP of 6.61% specifically for the 'Debris' category.

In terms of Segmentation performance, the model achieved an AP of 6.56% across all IoU thresholds, with a precision of 22.34% at IoU=0.50. At IoU=0.75, the precision decreased to 1.67%. The Average Recall (AR) at IoU=0.50:0.95 was 26.1%. Specifically for the 'Debris' category, the model attained an AP of 6.56%.

The evaluation results suggest moderate performance of the Mask R-CNN model on the test dataset. Notably, the model exhibits better performance in detecting larger debris instances compared to smaller ones. However, there remains room for improvement, particularly in achieving higher precision and recall rates across different categories and IoU thresholds. Further refinement of the model architecture, dataset augmentation, and exploration of alternative object detection frameworks could potentially enhance its efficacy in real-world scenarios.

The evaluation results for the Mask R-CNN model trained with 8 epochs and 2000 iterations show improvements in performance compared to the previous evaluation. The model achieved an Average precision (AP) of 7.88% for bounding box detection, with a notable precision of 25.23% at IoU=0.50. The Average Recall (AR) across all IoU thresholds from 0.50 to 0.95 was computed to be 29.4%, indicating enhanced detection capabilities, particularly for smaller debris instances. For Segmentation, the model attained an AP of 7.49%, with a precision of 25.71% at IoU=0.50. The AR at IoU=0.50:0.95 was 25.8%, with improved recall rates across all object sizes compared to the previous evaluation.

The comparison between the models trained with 4 and 1500 epochs versus 8 epochs and 2000 iterations indicates a noticeable improvement in performance with increased training epochs and iterations. Specifically, the model trained with 8 epochs and 2000 iterations demonstrates higher Average precision and Recall values for both bounding box detection and Segmentation tasks. This suggests that increasing the epochs and iterations has led to better model convergence and enhanced capability to detect debris instances across different sizes and complexities. Further analysis and experimentation may be warranted to optimize the model's performance and assess its robustness under varying conditions. Additionally, Figures 3.5 and 3.6 contain the images that underwent Segmentation applied to the test datasets.

| Metric | 4 Epochs, 1500 Iterations | 8 Epochs, 2000 Iterations |
|---|---|---|
| BBox AP | 6.61% | 7.88% |
| BBox AP50 | 2.78% | 22.04% |
| BBox AP75 | 3.99% | \| 25.23% |
| Segm AP | 6.56% | 7.49% |
| Segm AP50 | 22.34% | 25.71% |
| Segm AP75 | 1.67% | 2.48% |
| Average Recall | 28.9% | 29.4% |
| Small Debris AR | 37.5% | 35.0% |
| Medium Debris AR | 26.5% | 26.1% |
| Large Debris | 36.1% | 39.8% |

**Table 3.1** Performance comparisons between models trained for 4 epochs / 1500 iterations and 8 epochs / 2000 iterations

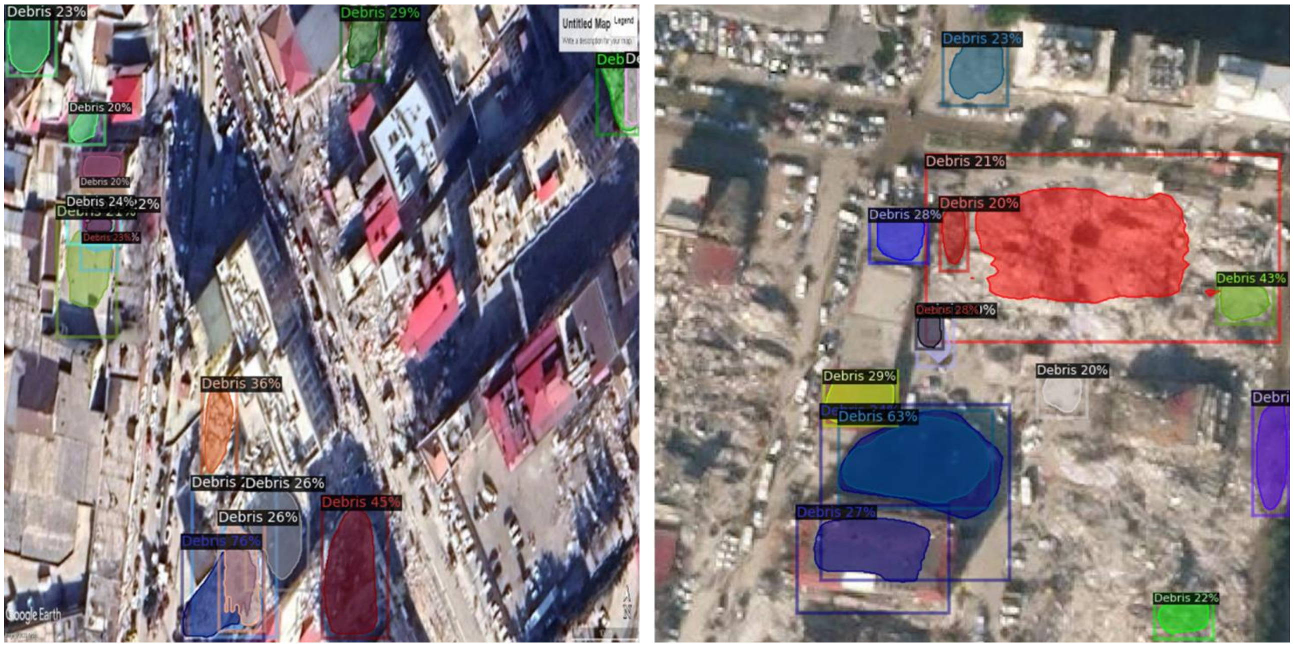


**Figure 3.45** Test Data Predictions Utilizing Mask R-CNN via Detectron2 with 4 Epochs and 1500 Iteration.

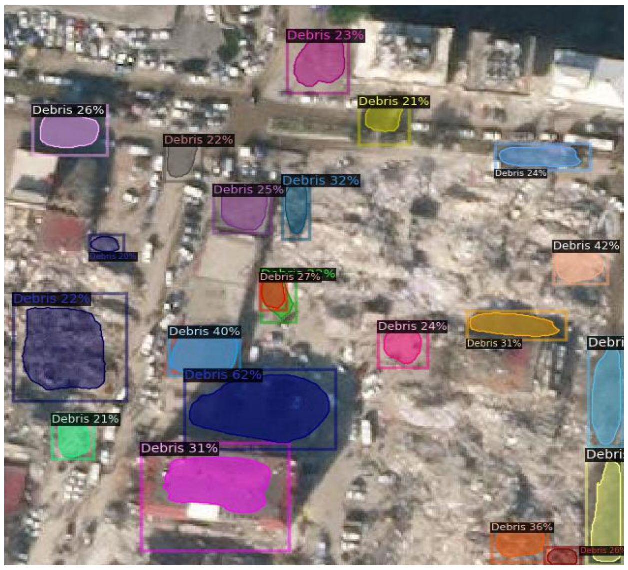


**Figure 3.46** Test Data Predictions Utilizing Mask R-CNN via Detectron2 with 8 Epochs and 2000 Iterations

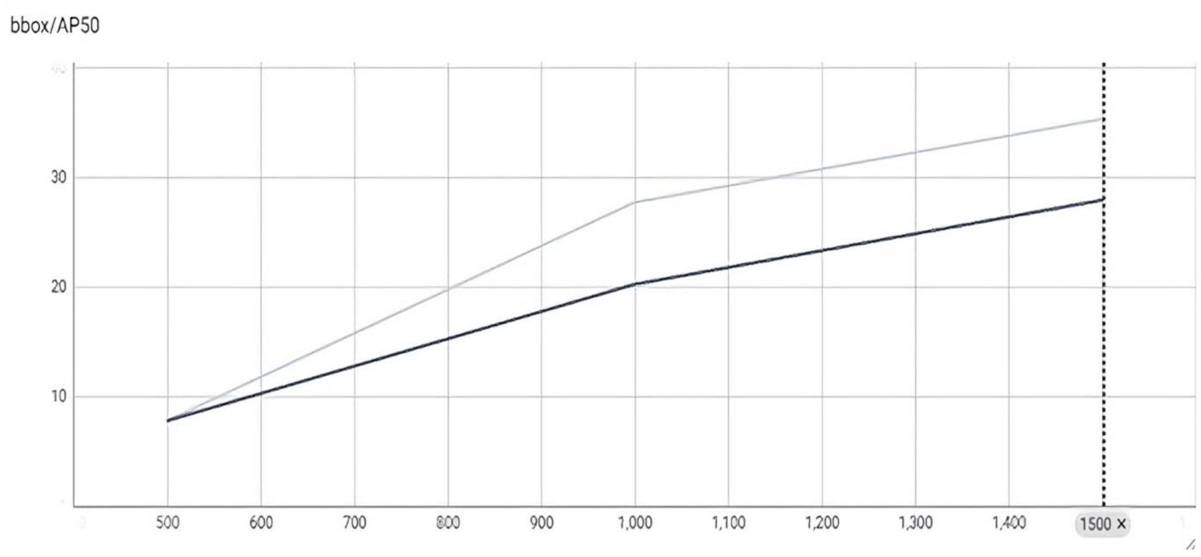


**Figure 3.47** Object Detection Performance using Mask R-CNN via Detectron2 after 4 Epochs and 1500 Iterations

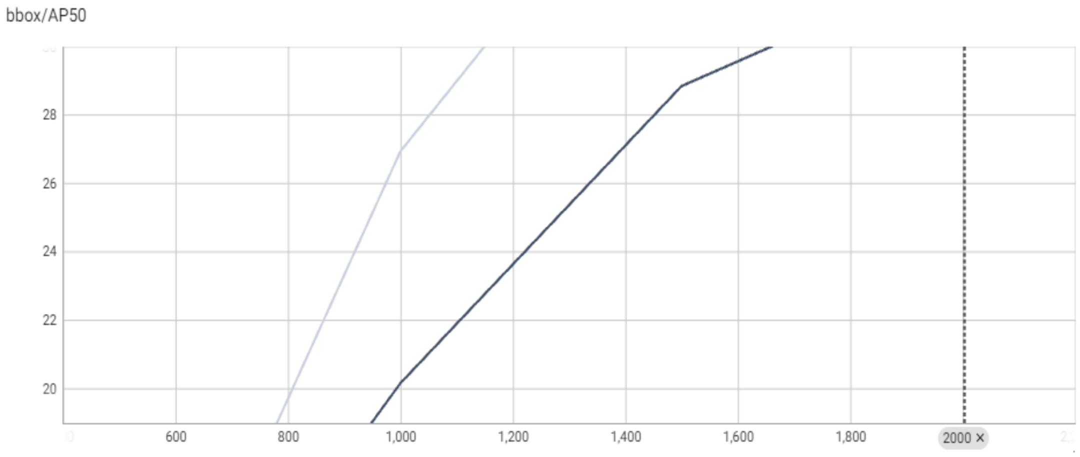


**Figure 3.48** Object Detection Performance using Mask R-CNN via Detectron2 after 8 Epochs and 2000 Iteration.

### 3.2.4. Roboflow Instance Segmentation Results

The instance Segmentation results obtained from Roboflow on my dataset demonstrates a moderate level of performance. With a mean Average precision (mAP) of 53.5%, the model shows an ability to accurately identify and delineate instances within the images. Precision, at 58.0%, indicates the proportion of correctly identified instances among all instances predicted by the model, reflecting its capability to minimize false positives. Meanwhile, the recall score the 53.9% demonstrates how well the model captures a sizable percentage of real positive events out of all positive instances. While these metrics indicate a reasonable performance level, there may still be room for improvement to enhance both precision and recall for better instance Segmentation results.

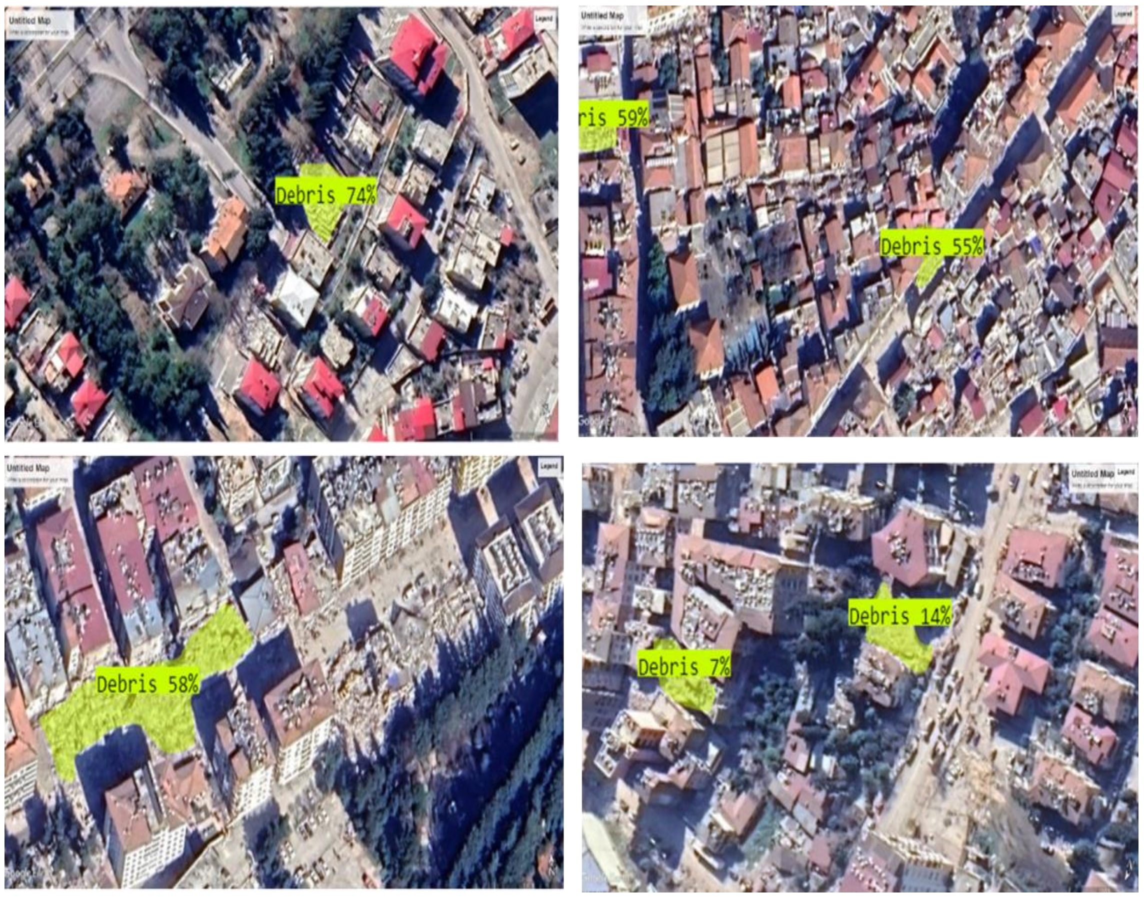


**Figure 3.49** A Visual Overview of satellite Dataset Analysis of Roboflow Instance Segmentation Results

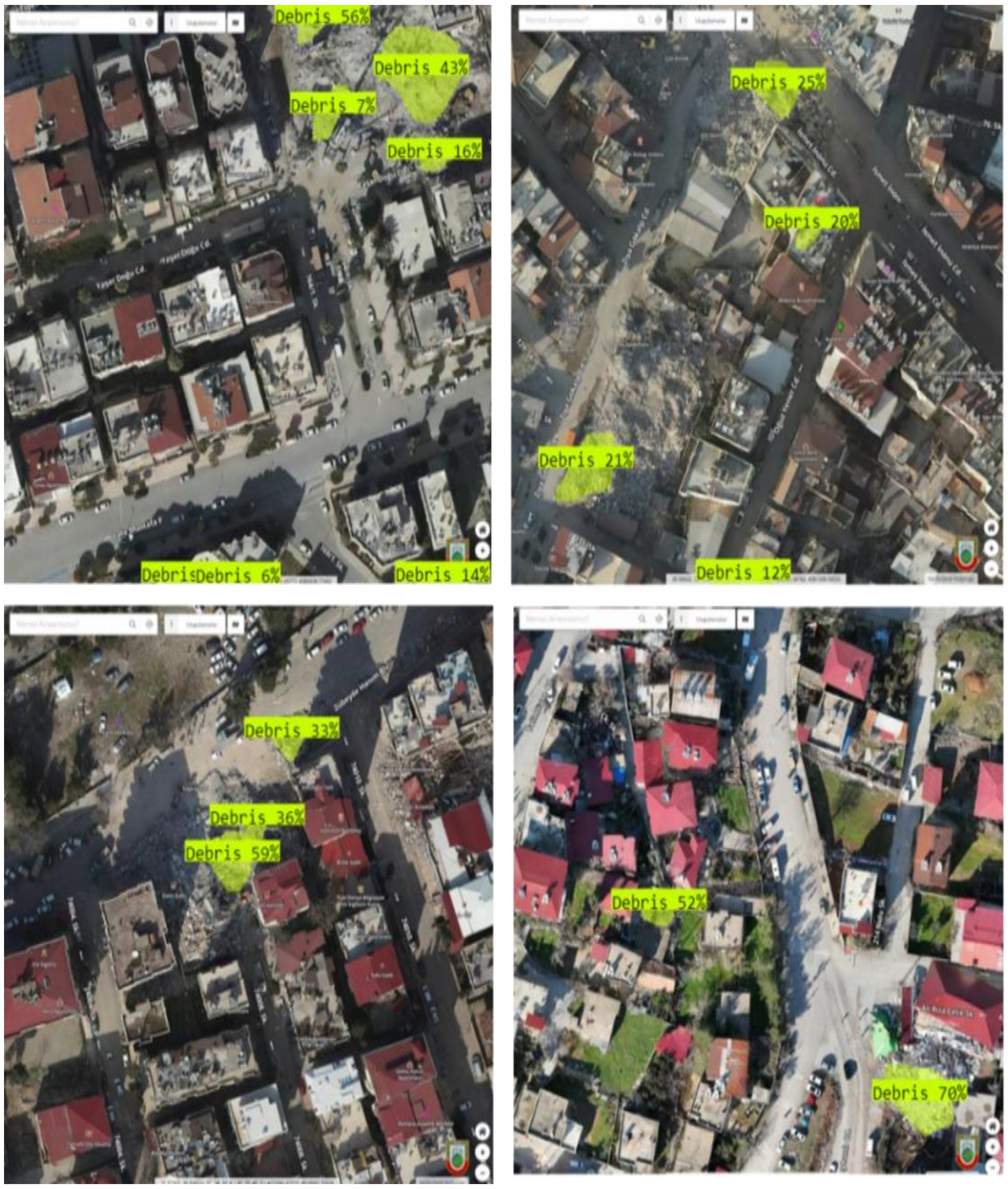


**Figure 3.50** A Visual Overview of drone based Dataset Analysis of Roboflow Instance Segmentation Results

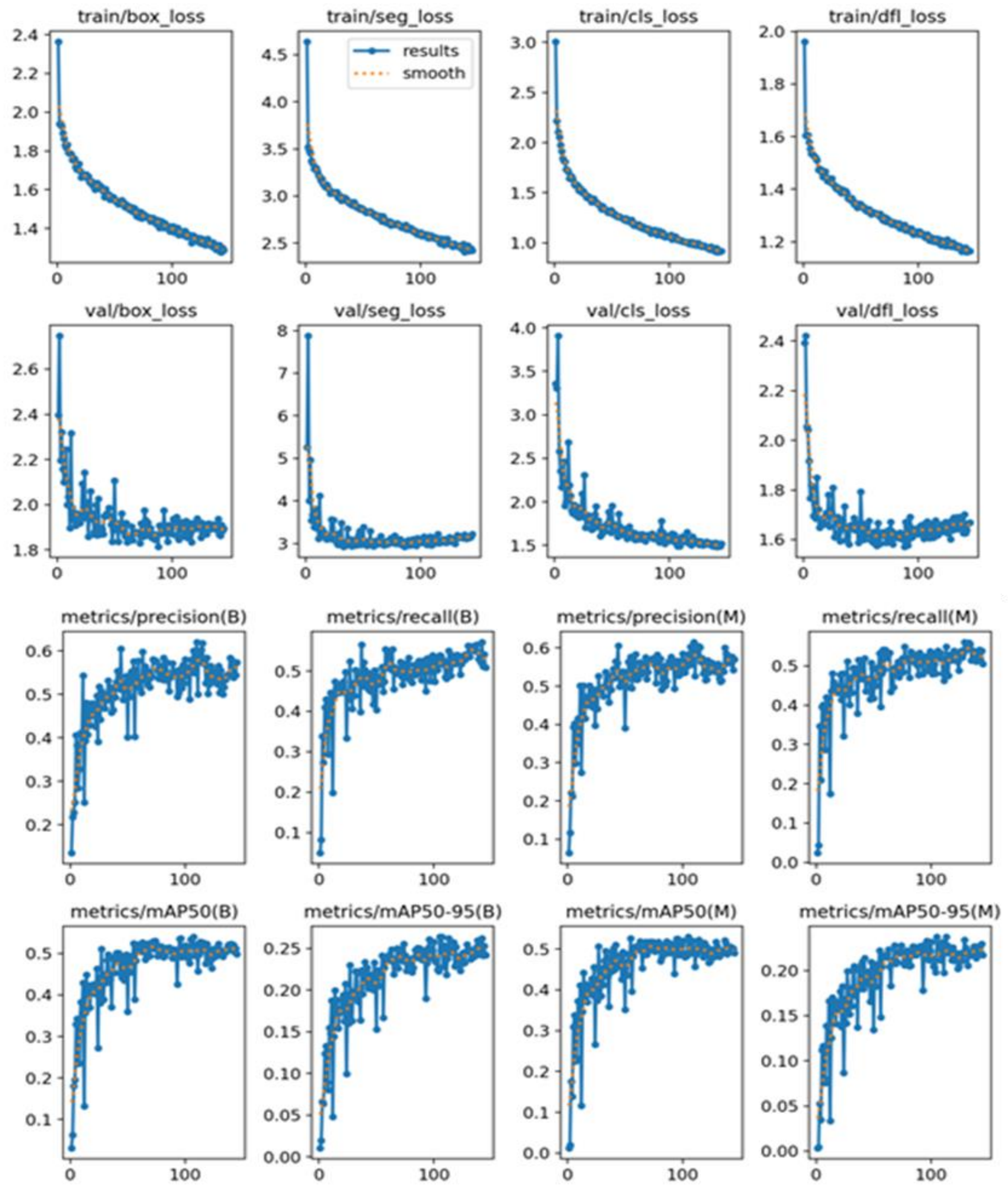


**Figure 3.51** Roboflow Instance Segmentation training Insights and Performance Metric.

## 3.3 Integrating Traffic Maps and Satellite Imagery for Comprehensive Analysis and Visualization

In this section, the integration of traffic maps and satellite imagery for comprehensive analysis and visualization is explored. The process began by downloading a map from an official source, specifically a TIF file named '63e6b164c6ef740006cfd110', depicting the city of

Kahramanmaras in Turkey following an earthquake event in 2023. This map was then imported into QGIS for further analysis. Within QGIS, images containing both the satellite imagery and the traffic layer were examined. Utilizing YOLOv8 Segmentation for debris Segmentation, predictions were made to identify areas of debris accumulation. Subsequently, the mask representing the debris was overlaid onto the traffic maps. This overlay facilitated the visualization of roadways and traffic patterns affected by the presence of debris, offering insights into the impact of the earthquake on transportation infrastructure and aiding in comprehensive analysis and decision-making processes.

Additionally, the TIF was accompanied by a JGW file, providing essential geo-referencing information. The JGW file contains parameters defining the spatial transformation and coordinates of the raster image, enabling its accurate positioning and scaling within GIS software. By leveraging the information from the JGW file, the satellite imagery depicting the city of Kahramanmaras was correctly georeferenced and integrated into the analysis within QGIS.

Moreover, with the generation of debris mask and traffic map overlay images for each segment of the city map, the ability to navigate through the urban landscape amidst debris blockages was significantly enhanced. By overlaying the debris mask onto the traffic map, areas affected by debris accumulation were visually highlighted, providing valuable insights into the extent of road blockages and traffic disruptions following the earthquake event.

Leveraging the geo referencing information provided by the JGW files associated with the satellite imagery, route calculation algorithms could accurately determine navigational paths between specified streets within the affected area. By considering the blocked routes due to debris accumulation, these routing algorithms could dynamically adjust the proposed paths to ensure safe and efficient navigation through the city's road network. This advanced routing functionality enabled emergency responders, city planners, and residents to identify alternative routes, avoid impassable roads, and optimize travel times, thereby facilitating effective disaster response and recovery efforts.

Furthermore, the integration of real-time data feeds and geospatial analytics allowed for the continuous monitoring and updating of navigation routes in response to changing conditions on the ground. Satellite imagery, drone footage, and crowd-sourced reports of debris clearance efforts were integrated into the GIS platform, providing up-to-date information on road conditions and accessibility. This dynamic, data-driven approach to urban navigation not only improved the safety and efficiency of travel but also enhanced situational awareness and coordination among stakeholders involved in disaster response and recovery operations.

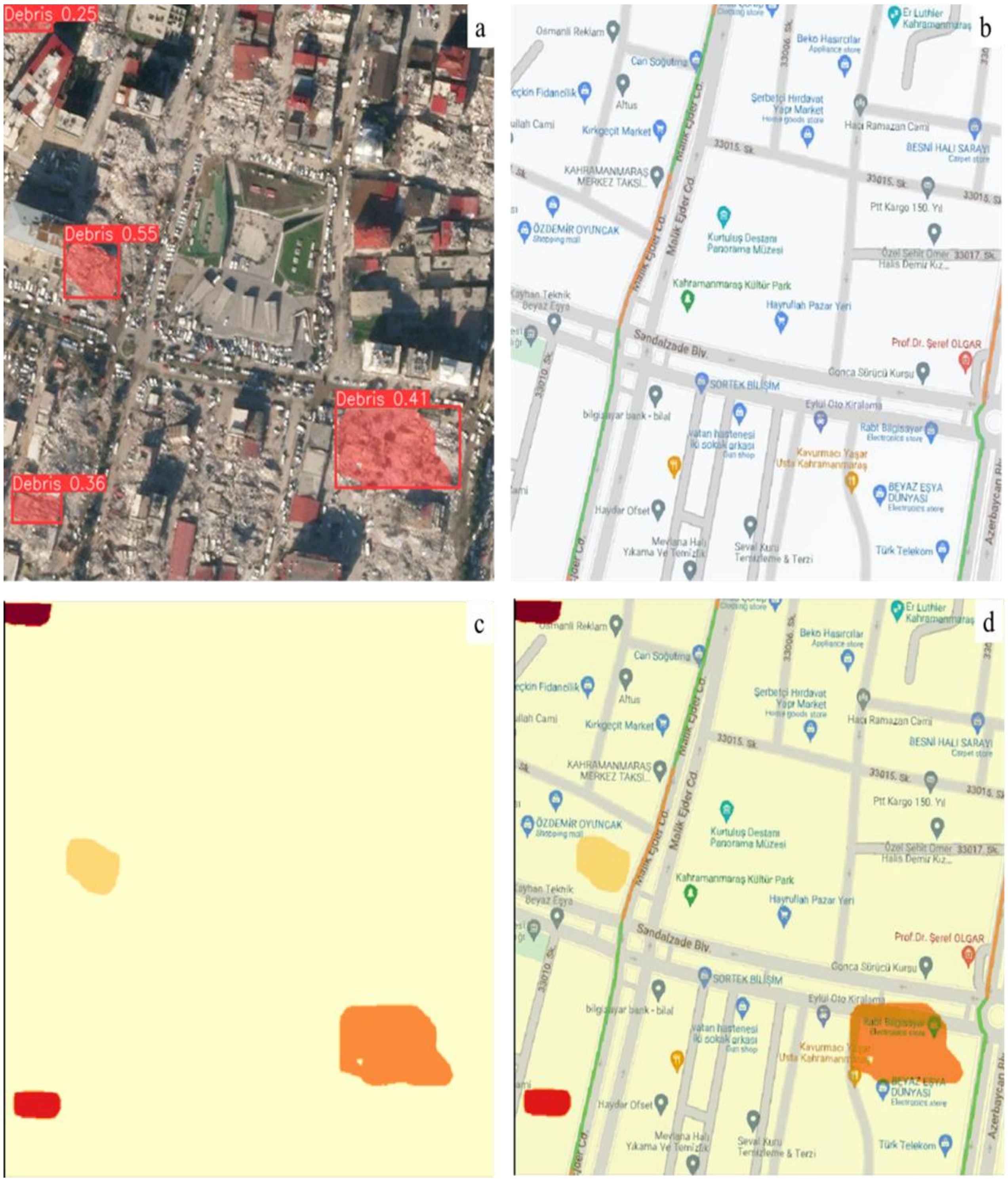


**Figure 3.52** a) YOLOv8 Predicted Debris Result for Coordinates 316485.1, 4161069,b) Associated Traffic Map, c) Mask Representation, d) Overlay of Mask and Traffic

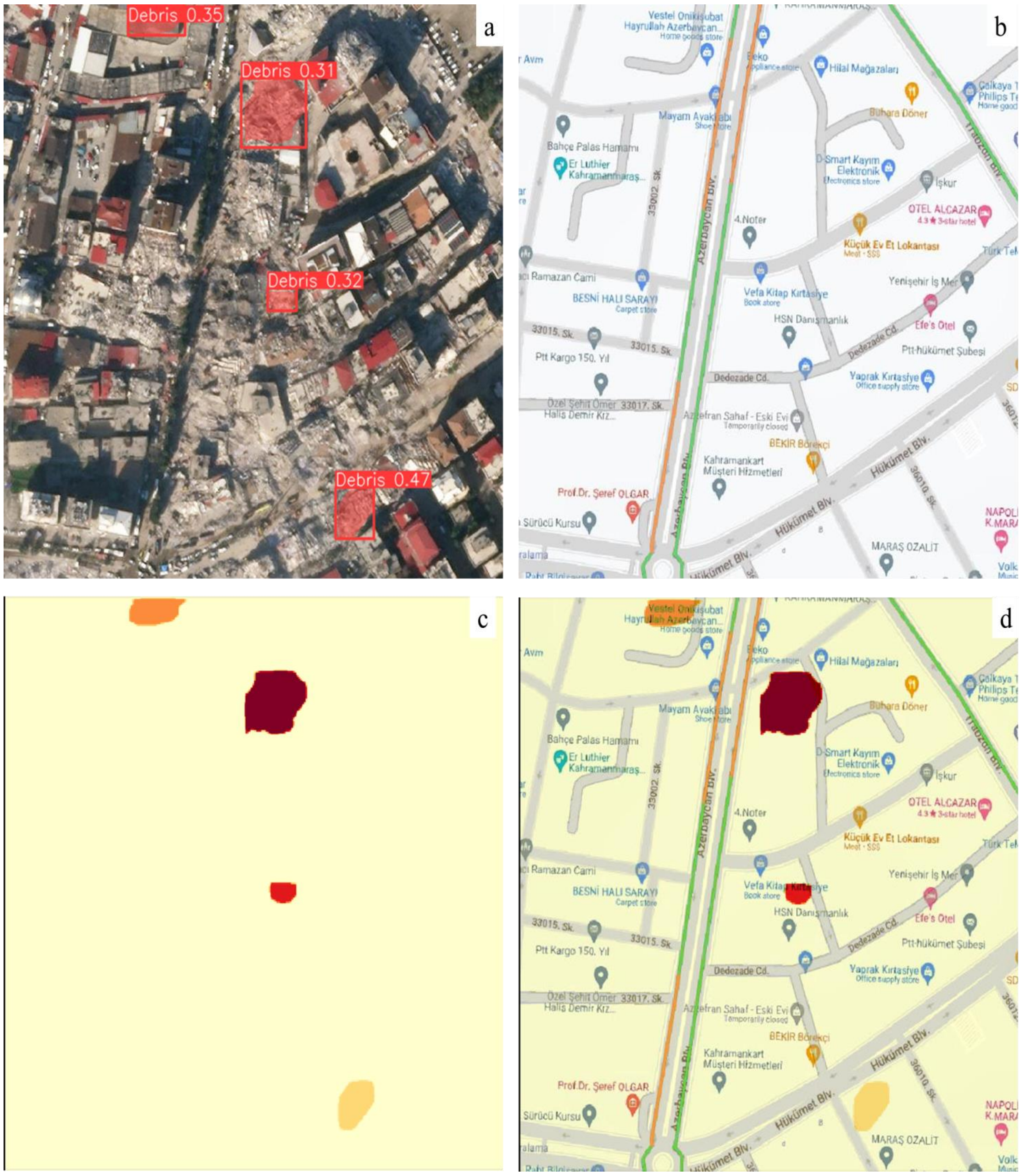


**Figure 0.53** a) YOLOv8 Predicted Debris Result for Coordinates 316717.3, 4161155,b) Associated Traffic Map, c) Mask Representation, d) Overlay of Mask and Traffic

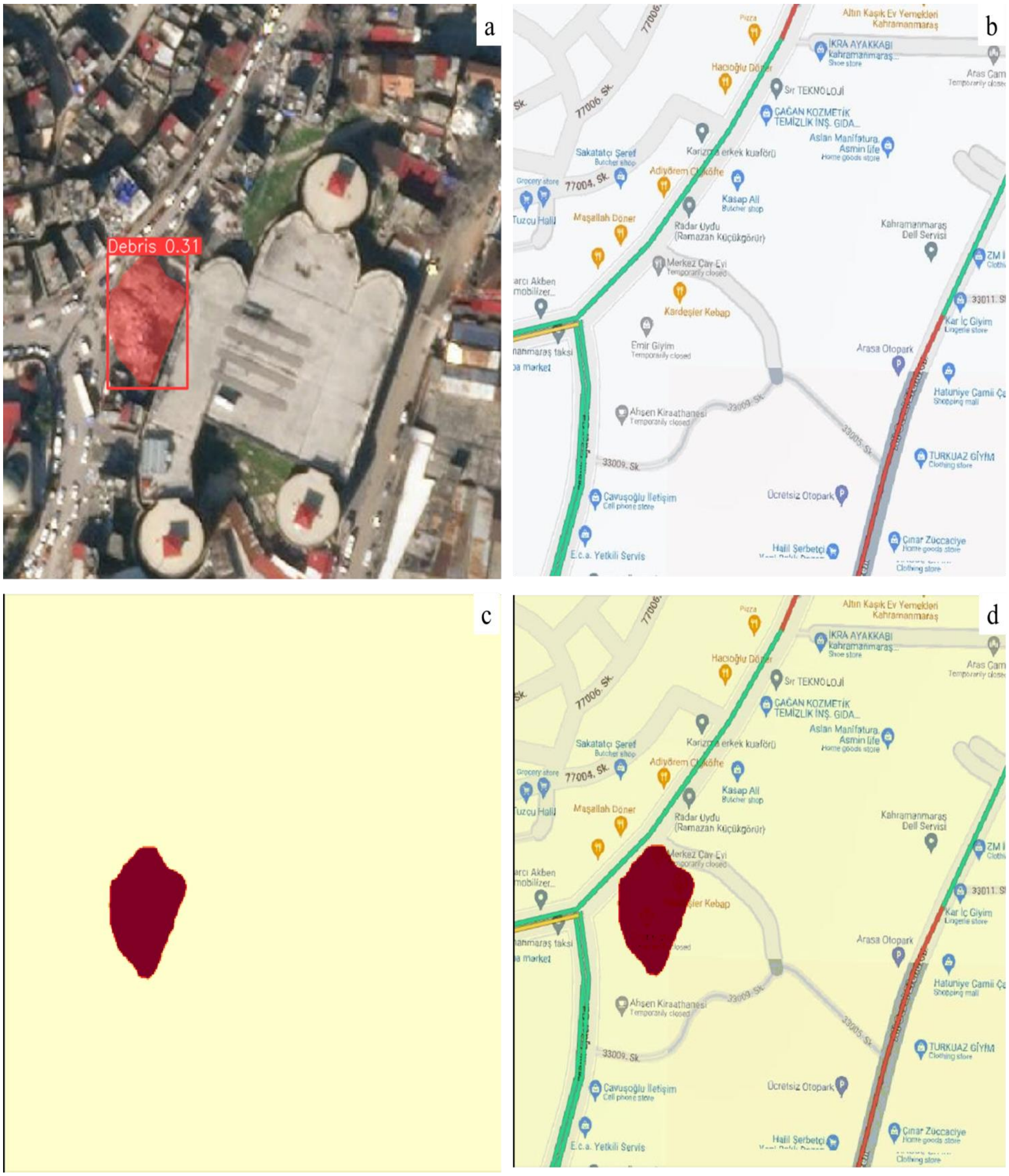


**Figure 0.54** a) YOLOv8 Predicted Debris Result for Coordinates 316682.9, 4161409.5,b) Associated Traffic Map, c) Mask Representation, d) Overlay of Mask and Traffic.

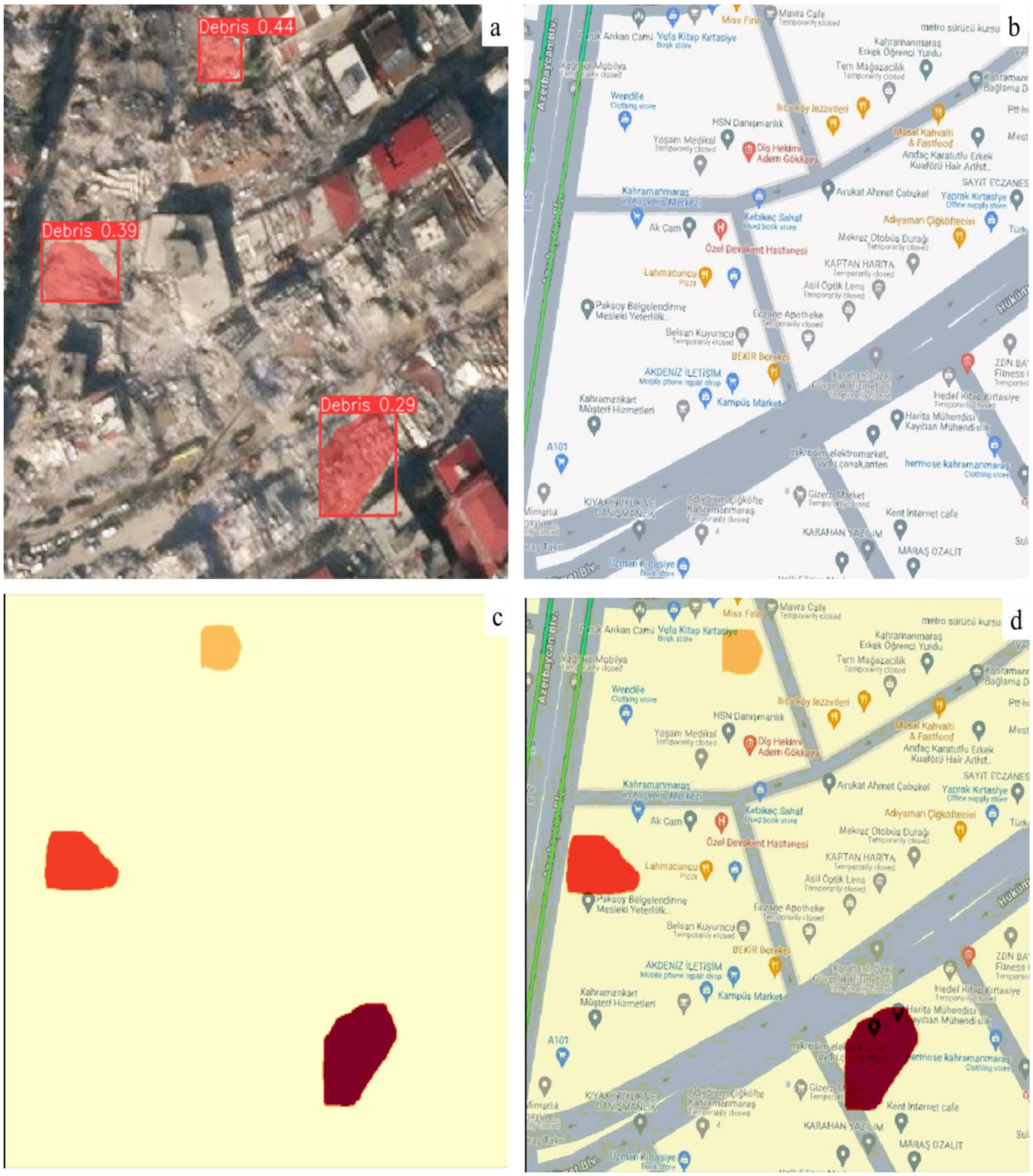


**Figure 0.55** a) YOLOv8 Predicted Debris Result for Coordinates 316877.6, 4161036.4, b) Associated Traffic Map, c) Mask Representation, d) Overlay of Mask and Traffic

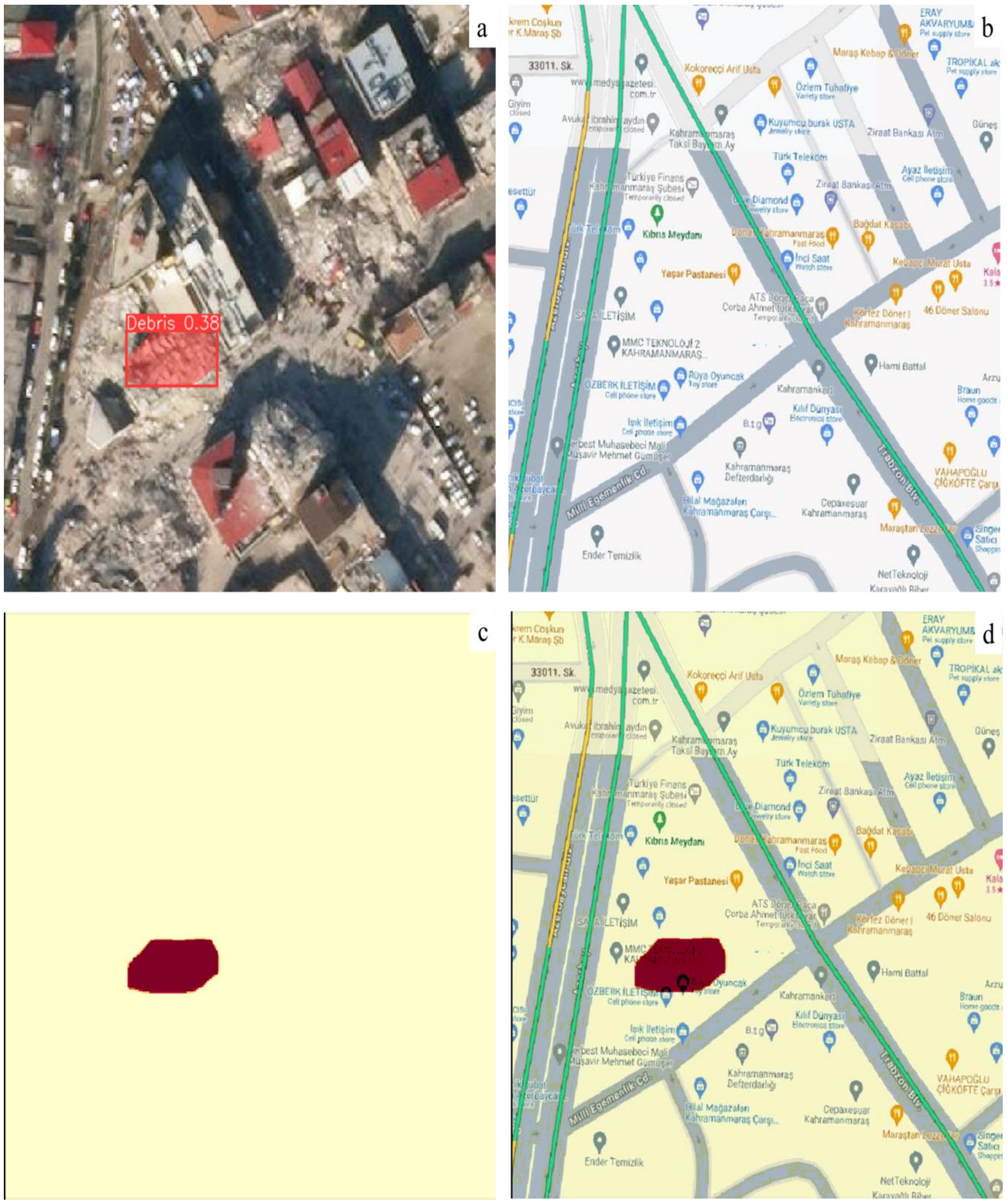


**Figure 0.56** a) YOLOv8 Predicted Debris Result for Coordinates 316895.3, 4161318, b) Associated Traffic Map, c) Mask Representation, d) Overlay of Mask and Traffic

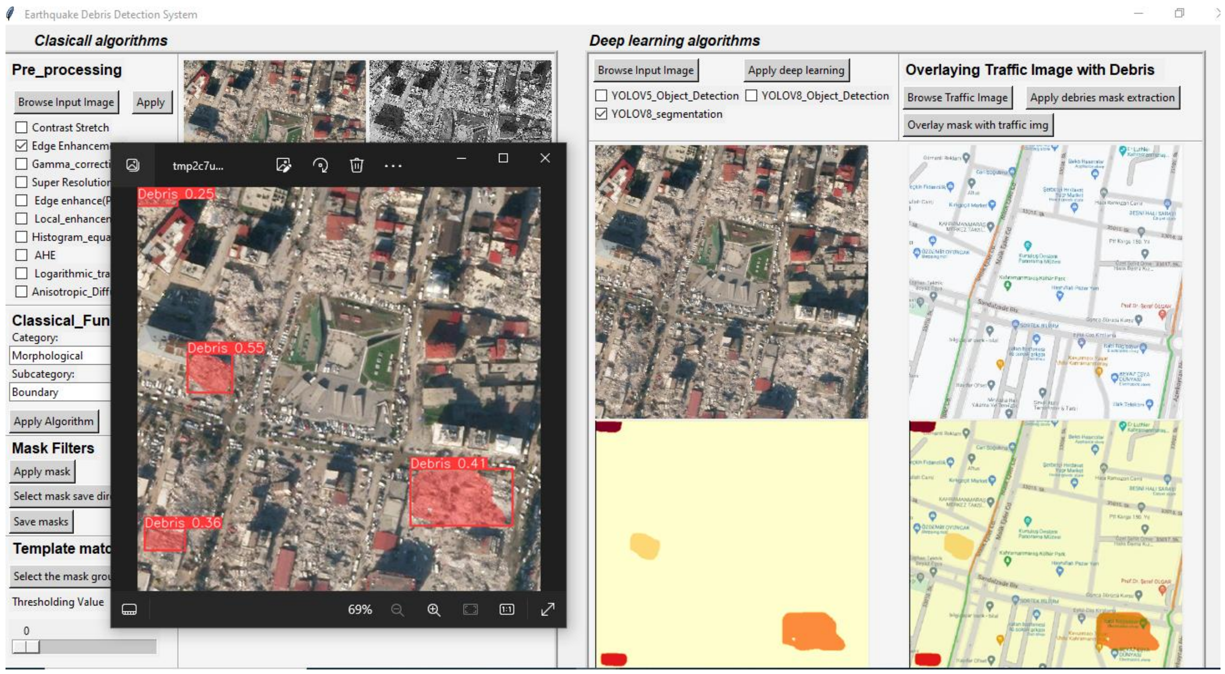


**Figure 3.57** Overlaying Debris Masks on Traffic Imagery within the GUI

# 4. CONCLUSIONS

In the initial stages of our research, we employed classical image processing algorithms to identify debris within drone-captured images following earthquakes. Through experimentation and analysis, we explored various traditional techniques to delineate debris regions and create masks. Among the nine algorithms tested, boundary detection emerged as particularly adept at discerning debris textures. However, while classical algorithms provided some insights, they fell short in accurately detecting debris due to the diverse nature of debris textures. We propose that increasing the diversity and number of masks could potentially enhance classical algorithm performance by better representing the varied debris textures. Nonetheless, this approach entails significant time investment, prompting consideration of alternative methods such as Deep learning, which offer greater accuracy and efficiency.

In our exploration of Deep learning algorithms for object detection and Segmentation tasks, we analyzed several state-of-the-art models, including YOLOv5, YOLOv8, Mask R-CNN with Detectron2, and Roboflow Instance Segmentation. Each algorithm's performance was meticulously assessed, with discussions on key metrics such as precision, recall, loss functions, and mean average precision (mAP), complemented by visual representations of dataset prediction images. Through this comprehensive evaluation, we aimed to gauge the suitability and efficacy of each algorithm in addressing our target tasks.

The evaluation of the YOLOv5m model yielded promising results in object detection, achieving a commendable precision of 56.45% and a recall of 63.03%, with a mean average precision (mAP) of 57.105%. However, precision levels fluctuated due to diverse debris textures. In contrast, YOLOv8 exhibited relatively higher losses in bounding box predictions and objectness scores but achieved moderate performance metrics. The YOLOv8 Segmentation model showed improvements in various metrics in the 200th epoch. The Mask R-CNN and Roboflow Instance Segmentation models demonstrated moderate performance in Segmentation, with better results for larger debris instances. Further enhancements are needed for precision and recall rates across different categories and thresholds.

In our investigation, we utilized traffic maps and satellite imagery to analyze post-earthquake scenarios. Through YOLOv8 Segmentation; we identified debris areas, overlaying them onto traffic maps to visualize affected roadways. By overlaying debris masks onto traffic maps, we improved navigation amidst debris, aiding in disaster response. Integration of real-time data feeds allowed continuous route monitoring and updating, enhancing situational awareness among stakeholders. This dynamic approach offers a valuable tool for urban navigation, ensuring safe travel and effective disaster management.

Furthermore, optimizing Segmentation results and developing a user-friendly application that operates effectively post-earthquake are crucial steps. Creating an application that functions reliably in the aftermath of an earthquake event is essential for facilitating efficient navigation and disaster response efforts.

For further optimization of my thesis and project, several advanced strategies can be employed. Enhancing the quality of our dataset remains a primary focus. By utilizing advanced data augmentation techniques, refining annotation processes, and collecting datasets with the best possible resolution, we aim to improve the model's accuracy. Addressing the challenge of partially and completely blocked roads is also critical, enabling us to define optimal routes for post-earthquake disaster assistance. Incorporating video processing capabilities into our workflow is essential for analyzing dynamic visual data, allowing for comprehensive insights into temporal patterns and changes over time. Additionally, integrating an online Geographic Information System (GIS) platform can facilitate real-time disaster detection and management post-earthquake by providing up-to-date spatial data. Reducing computational time is another key factor; optimizing algorithms and streamlining processes will significantly decrease the time required for model training and inference, enhancing productivity. Moreover, considering the elevation of imagery and camera is crucial, as the model struggles with detecting debris when the camera elevation varies significantly. By incorporating images taken from different elevations, we can help the model understand debris patterns more accurately. Collectively, these strategies aim to enhance the system's efficiency, accuracy, and real-time applicability, ultimately improving disaster response and management.

# REFERENCES

[1] Karji, A., Woldesenbet, A., & Rokooei, S. (2017). Integration of augmented reality, building information modeling, and image processing in construction management: A content analysis. In Proceedings of the AEI 2017. American Society of Civil Engineers.

[2] Yu, L., He, S., Liu, X., Jiang, S., & Xiang, S. (2022). Intelligent crack detection and quantification in the concrete bridge: A Deep learning-assisted image processing approach. Advances in Civil Engineering, 2022(1813821), 1-15.

[3] Fukuda, Y., Feng, M. Q., & Shinozuka, M. (2010). Cost-effective vision-based system for monitoring dynamic response of civil engineering structures. Structural Control and Health Monitoring, 17(8), 918–936.

[4] Ramachandran, R. M., & Reddy, C. S. (2017). Monitoring of deforestation and land use changes (1925-2012) in Idukki district, Kerala, India using remote sensing and GIS. Journal of the Indian Society of Remote Sensing, 45(1), 163–170.

[5] Bohn, J. S., & Teizer, J. (2010). Benefits and barriers of construction project monitoring using high-resolution automated cameras. Journal of Construction Engineering and Management, 136(6), 632–640.

[6] Schindler, S., Hegemann, F., Koch, C., König, M., & Mark, P. (2016). Radar interferometry based settlement monitoring in tunnelling: visualisation and accuracy analyses. Visualization in Engineering, 4.

[7] Vasileva, A. V., Vasilev, A. S., & Konyakhin, I. A. (2018). Vision-based system for long-term remote monitoring of large civil engineering structures: design, testing, evaluation. Measurement Science and Technology, 29(11), Article ID 115003.

[8] Bashir, H., & Ahmad, S. S. (2017). Exploring geospatial techniques for spatiotemporal change detection in land cover dynamics along Soan River, Pakistan. Environmental Monitoring and Assessment, 189(5), 222.

[9] Ehrhart, M., & Werner, L. (2015, February). Image-based dynamic deformation monitoring of civil engineering structures from long ranges. In Proceedings of the Image Processing: Machine Vision Applications VIII, SPIE, San Francisco, CA, USA. [SPIE]

[10] Yang, J., Park, M.-W., Vela, P. A., & Golparvar-Fard, M. (2015). Construction performance monitoring via still images, time-lapse photos, and video streams: now,

tomorrow, and the future. Advanced Engineering Informatics, 29(2), 211–224. [Elsevier]

[11] Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2015). Automated progress monitoring using unordered daily construction photographs and IFC-based building information models. ASCE, Journal of Computing in Civil Engineering, 29(1).

[12] Woldesenbet, A., Jeong, H. D., & Park, H. (2016). Framework for integrating and assessing highway infrastructure data. Journal of Management in Engineering, 32(1), Article ID 04015028.

[13] Patel, K. K., Kar, A., Jha, S. N., & Khan, M. A. (2011). Machine vision system: a tool for quality inspection of food and agricultural products. Journal of Food Science & Technology, 49(2), 123–141.

[14] Xu, G., Deng, M., Sun, G., Guo, Y., & Chen, J. (2022). Improving Building Extraction by Using Knowledge Distillation to Reduce the Impact of Label Noise. Remote Sensing, 14(22), 5645.

[15] Salunkhe, A. A., Gobinath, R., Vinay, S., & Joseph, L. (2022). Progress and Trends in Image Processing Applications in Civil Engineering: Opportunities and Challenges. Volume 2022, Article ID 6400254.

[16] Gonzalez, R. C., & Woods, R. E. (2008). Digital image processing, Third Edition.

[17] Dey, S. (2018). Hands-On image processing with Python.

[18] Ferrer, J., Pomares, J. C., Irles, R., Espinosa, J., & Mas, D. (2013). image processing for safety assessment in civil engineering. Applied Optics, 52(18), 4385.

[19] Chen, L. Y., Wang, M. Y., Chao, C. C., & Lo, W. (2015). Assessment of asphalt concrete pavement quality by using infrared thermal imaging technology. Journal of Marine Science and Technology, 23(3), 331–338.

[20] dzgold. (2021). Deep learning Collection PDF. Retrieved from https://archive.org/details/deep-learning-collection-pdf

[21] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. Insights into Imaging, 9, 611–629.

[22] Salem, F. M. (2022). Recurrent neural networks: From Simple to Gated Architectures. Textbook.

[23] Tripathi, S., Augustin, A. I., Dunlop, A., Sukumaran, R., Dheer, S., Zavalny, A., Haslam, O., Austin, T., Donchez, J., Tripathi, P. K., & Kim, E. (2022). Recent advances and application of generative adversarial networks in drug discovery,

development, and targeting. Artificial Intelligence in Life Sciences. https://doi.org/10.1016/j.ailsci.2022.100045

[24] Long Short-Term Memory Network, Expert Systems with Applications, 2021.

[25] Li, P., Pei, Y., & Li, J. (2023). A comprehensive survey on design and application of autoencoder in Deep learning. Applied Soft Computing, 137, 110176. https://doi.org/10.1016/j.asoc.2023.110176 [26] Transfer learning: a friendly introduction, Journal of Big Data

[27] Niu, Z., Zhong, G., & Yu, H. (2021). A review on the attention mechanism of Deep learning. Neurocomputing, 460, 381-401. https://doi.org/10.1016/j.neucom.2021.03.091

[28] Haq, M. U., Sethi, M. A. J., & Rehman, A. U. (2023). Capsule Network with Its Limitation, Modification, and Applications—A Survey. Machine Learning and Knowledge Extraction, 5(3), 891-921. https://doi.org/10.3390/make5030047

[29] 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, 53. https://doi.org/10.1186/s40537-021-00444-8

[30] 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, 53. https://doi.org/10.1186/s40537-021-00444-8

[31] Bai, C., Bai, X., & Wu, K. (Year). A Review: Remote Sensing Image Object Detection Algorithm Based on Deep learning. Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou 730070

[32] Szeliski, R. (Year). computer vision: Algorithms and Applications. In D. Gries & F. B. Schneider (Eds.), Texts in Computer Science. Springer. Available at: www.springer.com/series/3191.

[33] Park, S.C., Park, M.K., & Kang, M.G. (2003). Super-resolution image reconstruction: a technical overview. IEEE Signal Processing Magazine, 20(3), 21-36. doi:10.1109/MSP.2003.1203207

[34] Sonka, M., Hlavac, V., & Boyle, R. (2008). image processing, Analysis, and Machine Vision (3rd Edition). Brooks/Cole Publishing Co. [Source: DBLP] [Available at: https://www.researchgate.net/publication/220695728]

[35] Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J.B., & Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. computer vision, Graphics, and image processing, 39(3), 355-368. https://doi.org/10.1016/S0734-189X(87)80186-X

[36] Jain, A. K. (Year). Fundamentals of Digital image processing.

[37] Pratt, William K. (2007). Digital image processing: PIKS Scientific inside. (4th ed.). A Wiley-Interscience publication. ISBN: 978-0-471-76777-0

[38] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süssstrunk, S. (2012). SLIC superpixels. School of Computer and Communication Sciences (IC), École Polytechnique Fédérale de Lausanne (EPFL).

[39] Jardim, S., António, J., & Mora, C. (2023). Image thresholding approaches for medical image Segmentation - short literature review. Procedia Computer Science. https://doi.org/10.1016/j.procs.2023.01.439

[40] Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. IEEE Transactions on Systems, Man, and Cybernetics, 9(1), 62-66. https://doi.org/10.1109/TSMC.1979.4310076

[41] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. University of Washington, Allen Institute for AI, Facebook AI Research. Retrieved from http://pjreddie.com/yolo/

[42] Sary, I. P., Armin, E. U., & Andromeda, S. (2023). Performance Comparison of YOLOv5 and YOLOv8 Architectures in Human Detection Using Aerial Images. Electrical Engineering, ISSN 2355-3286.

[43] Reis, D., Kupec, J., Hong, J., Daoudi, A. (Year). Real-Time Flying Object Detection with YOLOv8. Georgia Institute of Technology.

[44] Lakshmanan, V., Görner, M., Gillard, R. (2021). Practical Machine Learning for computer vision. Released in July 2021.

[45] Sapkota, R., Ahmed, D., Karkee, M. (Year). Comparing YOLOv8 and Mask RCNN for object Segmentation in complex orchard environments. Center for precision & Automated Agricultural Systems, Washington State University, 24106 N Bunn Rd, Prosser, 99350, Washington, USA. [46] Deep learning for Computer Vision by Rajalingappaa Shanmugamani

[46] Rosebrock, A. (2017). Deep learning for Computer Vision with Python: Practitioner Bundle (1st Edition, 1.2.1). First printing, September 2017.

[47] https://github.com

[48] He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. In 2017 IEEE International Conference on Computer Vision. IEEE.

[49] Zhang, J. K., Fanous, M., Sobh, N., Kajdacsy-Balla, A., & Popescu, G. (Year). Automatic Colorectal Cancer Screening Using Deep learning in Spatial Light Interference Microscopy Data.

[50] https://docs.roboflow.com/datasets/adding-data

[51] Ronneberger, O., Fischer, P., & Brox, T. (Year). U-Net: Convolutional Networks for Biomedical Image Segmentation. Medical Image Computing and Computer-Assisted Intervention – MICCAI.

[52] Siddique, N., Paheding, S., Elkin, C. P., & Devabhaktuni, V. (Year). U-Net and Its Variants for Medical Image Segmentation: A Review of Theory and Applications.

## APPENDIX

### Appendix 1 – Images from YOLO Object Detection Batch and Validation Dataset Results

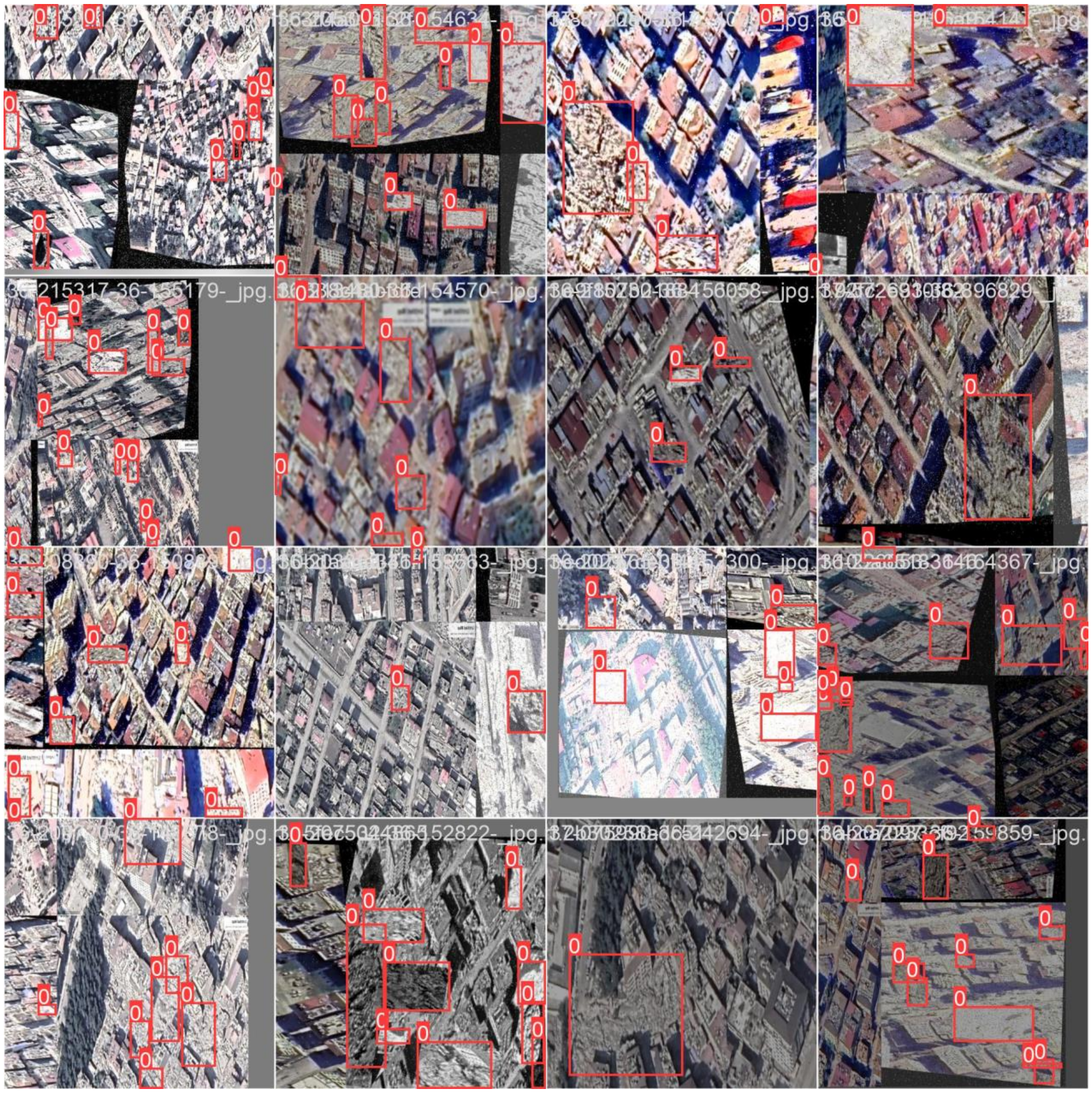


**Figure 1**YOLOv5m object detection Train batch number one

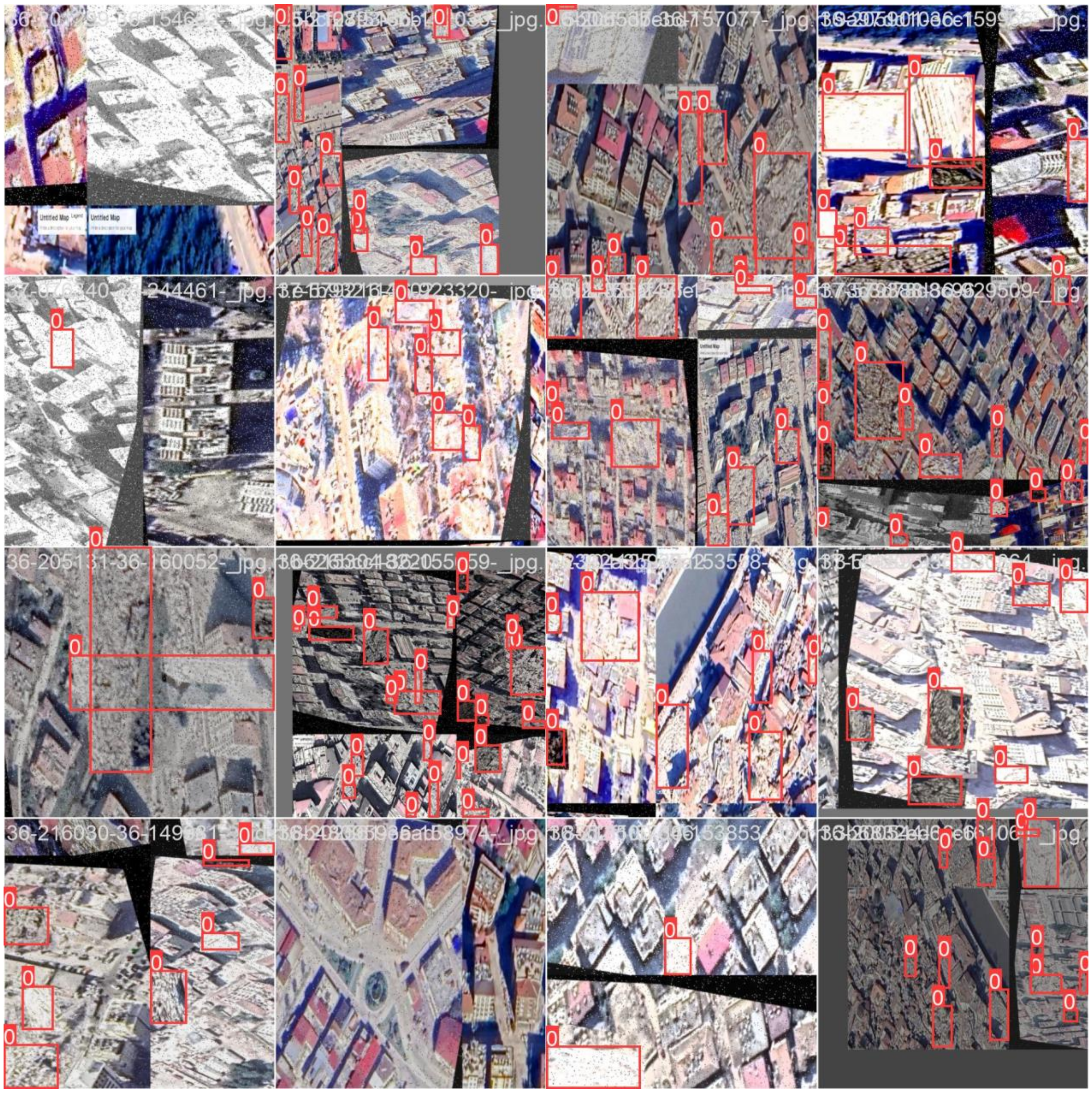


**Figure 2** YOLOv5m object detection train batch number two

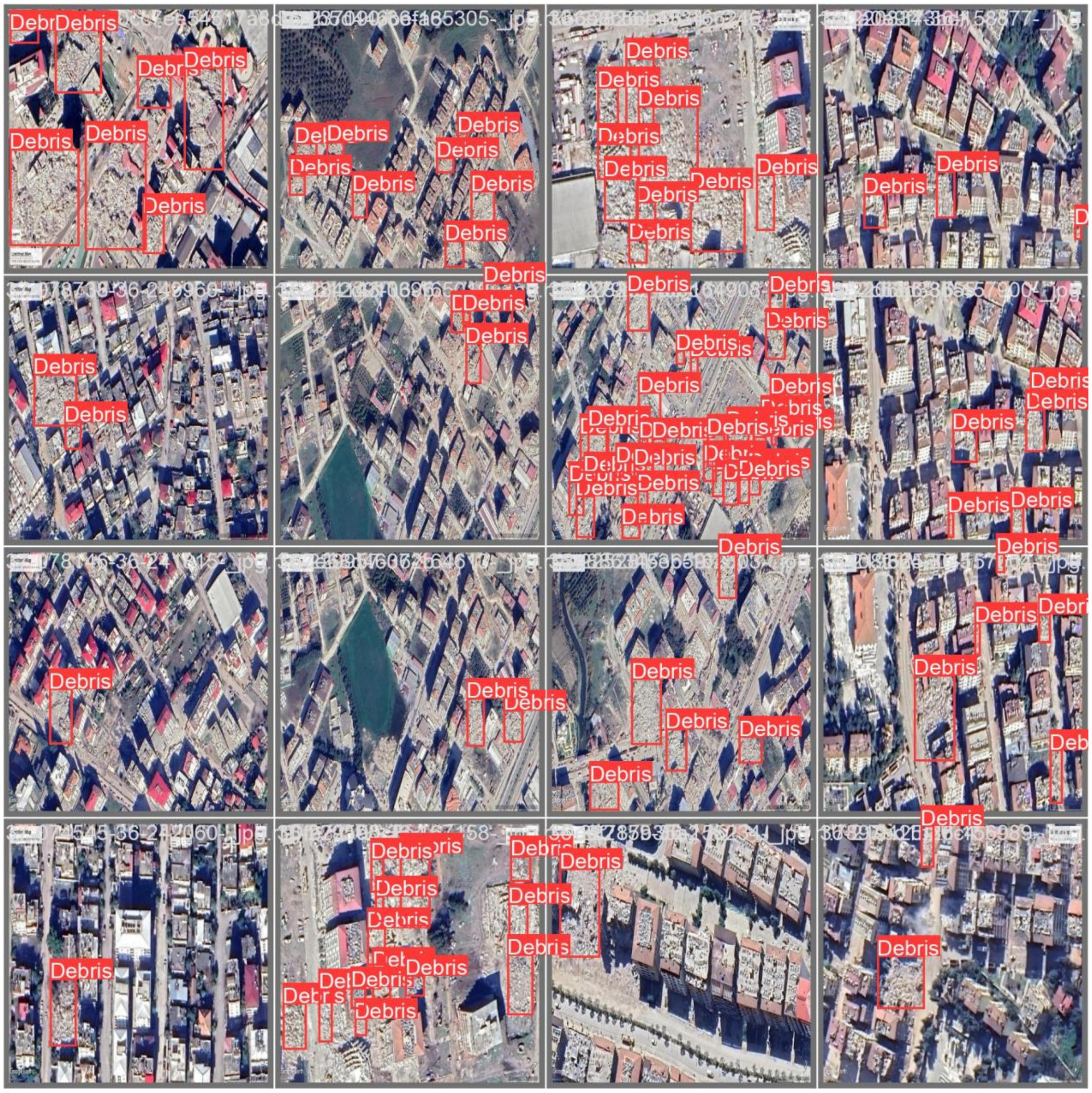


**Figure 3** YOLOv5m object detection validation batch number zero labels

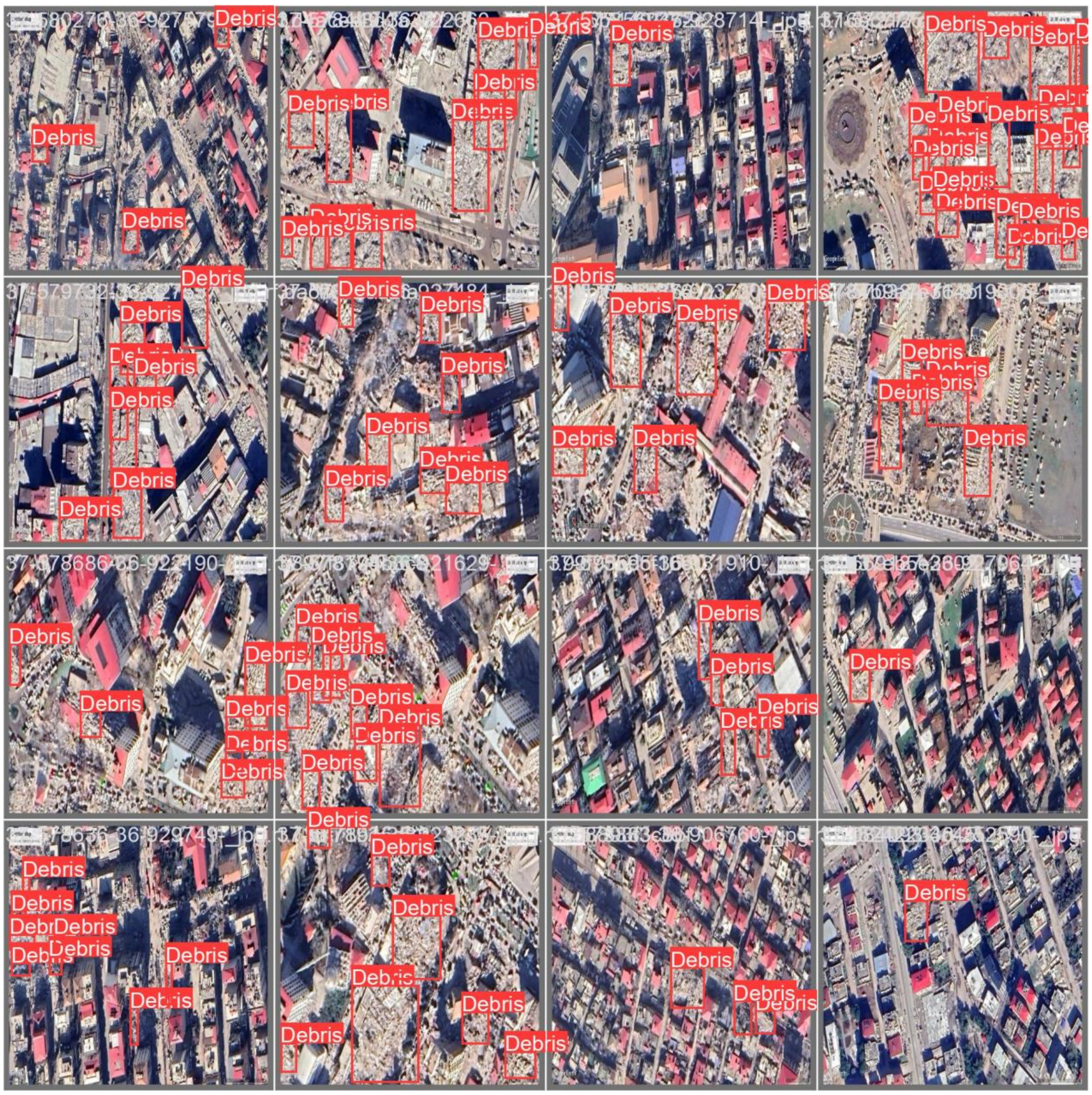


**Figure 4** YOLOv5m object detection validation batch number one labels

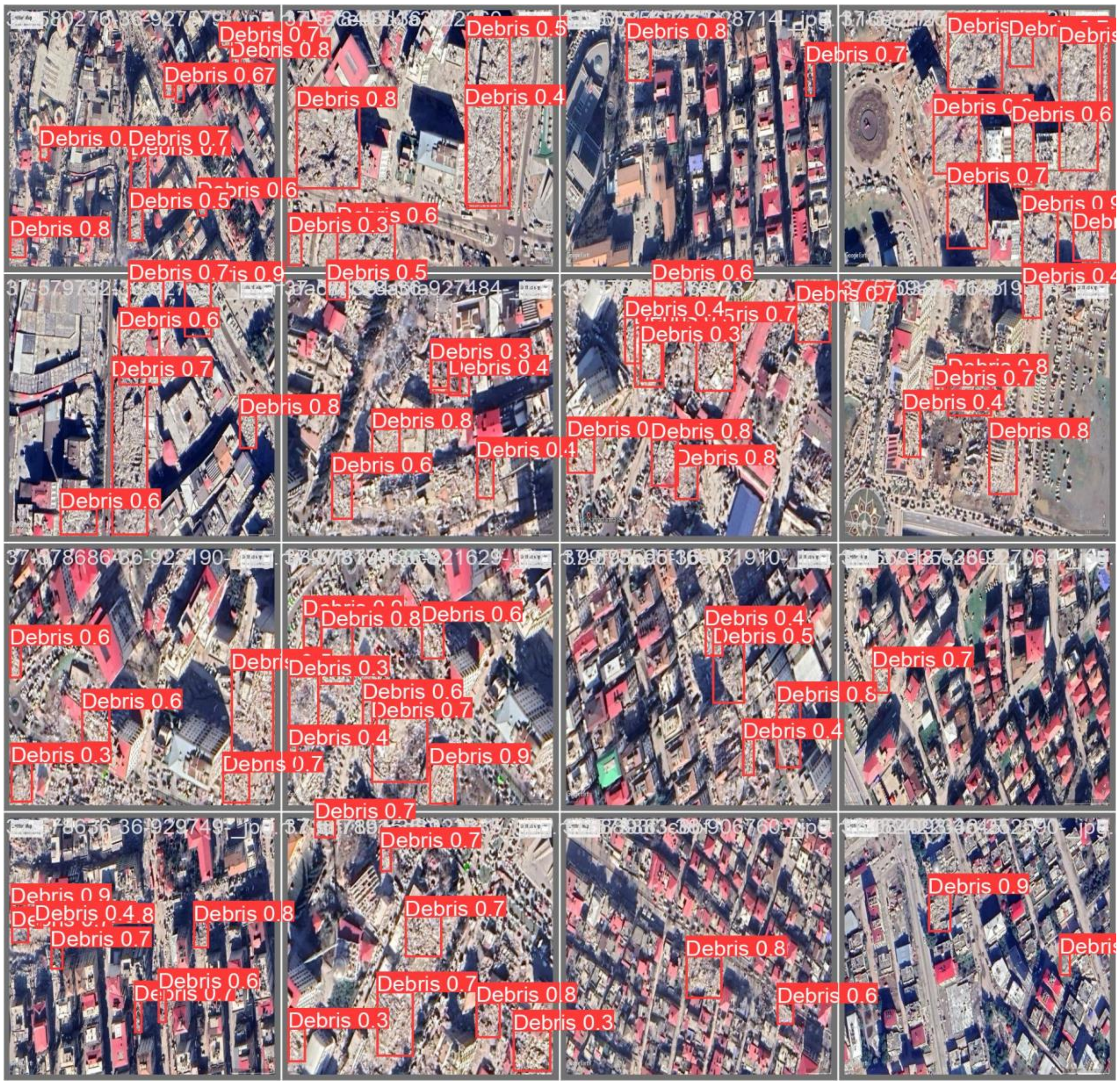


**Figure 5** YOLOv5m object detection validation batch number one prediction

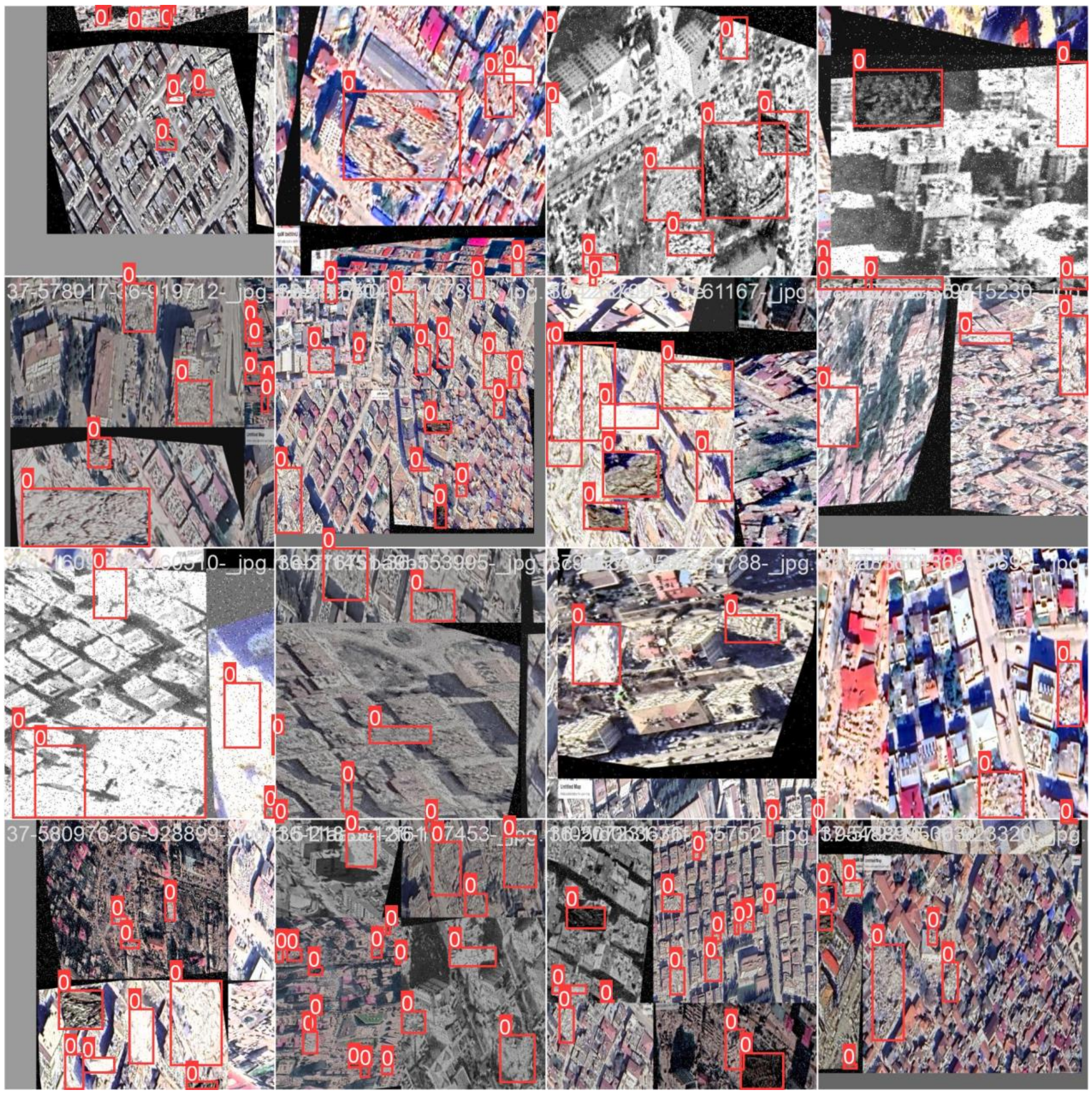


**Figure 6** YOLOv8 object detection train batch number zero

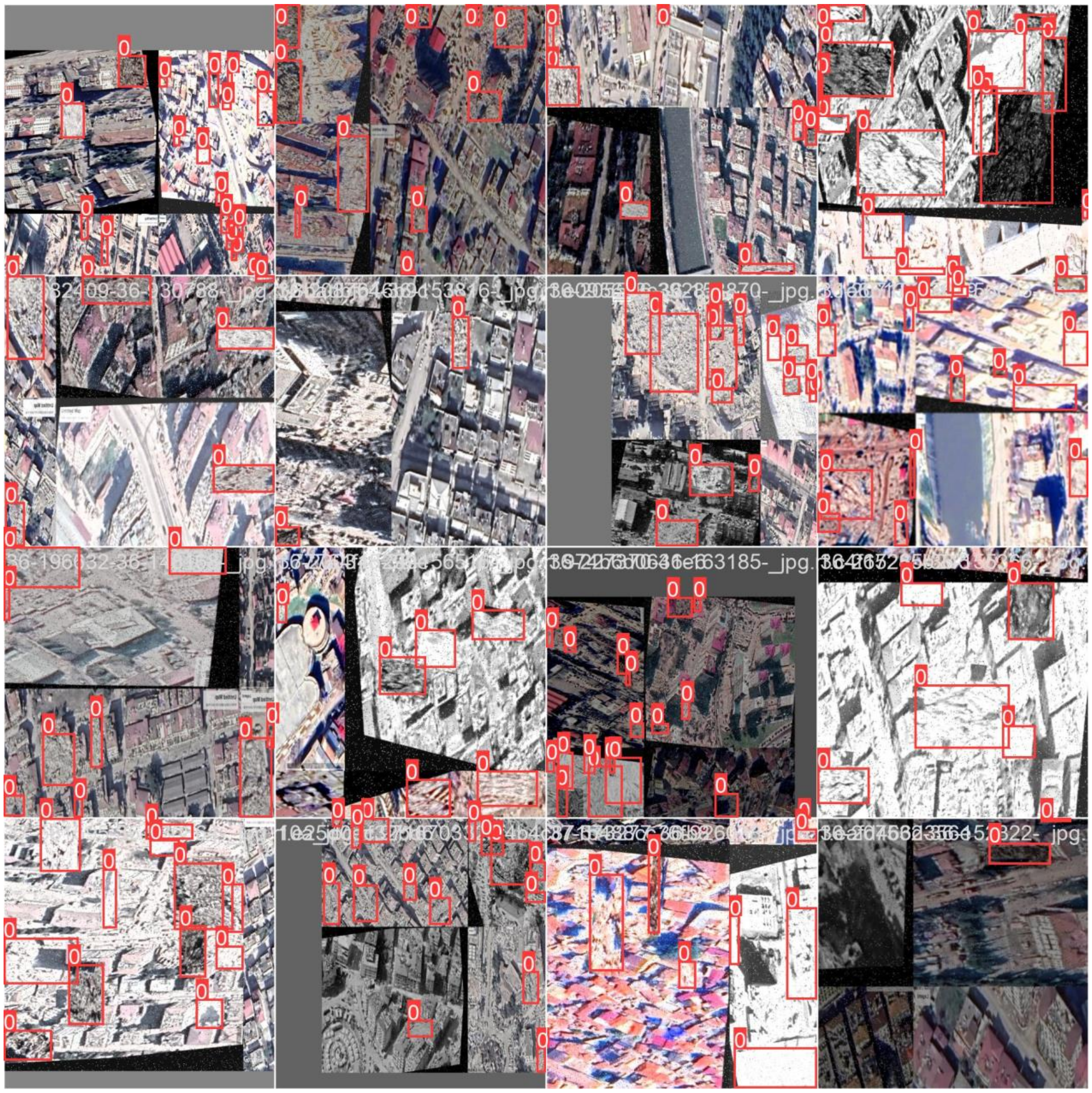


**Figure 7** YOLOv8 object detection train batch number one

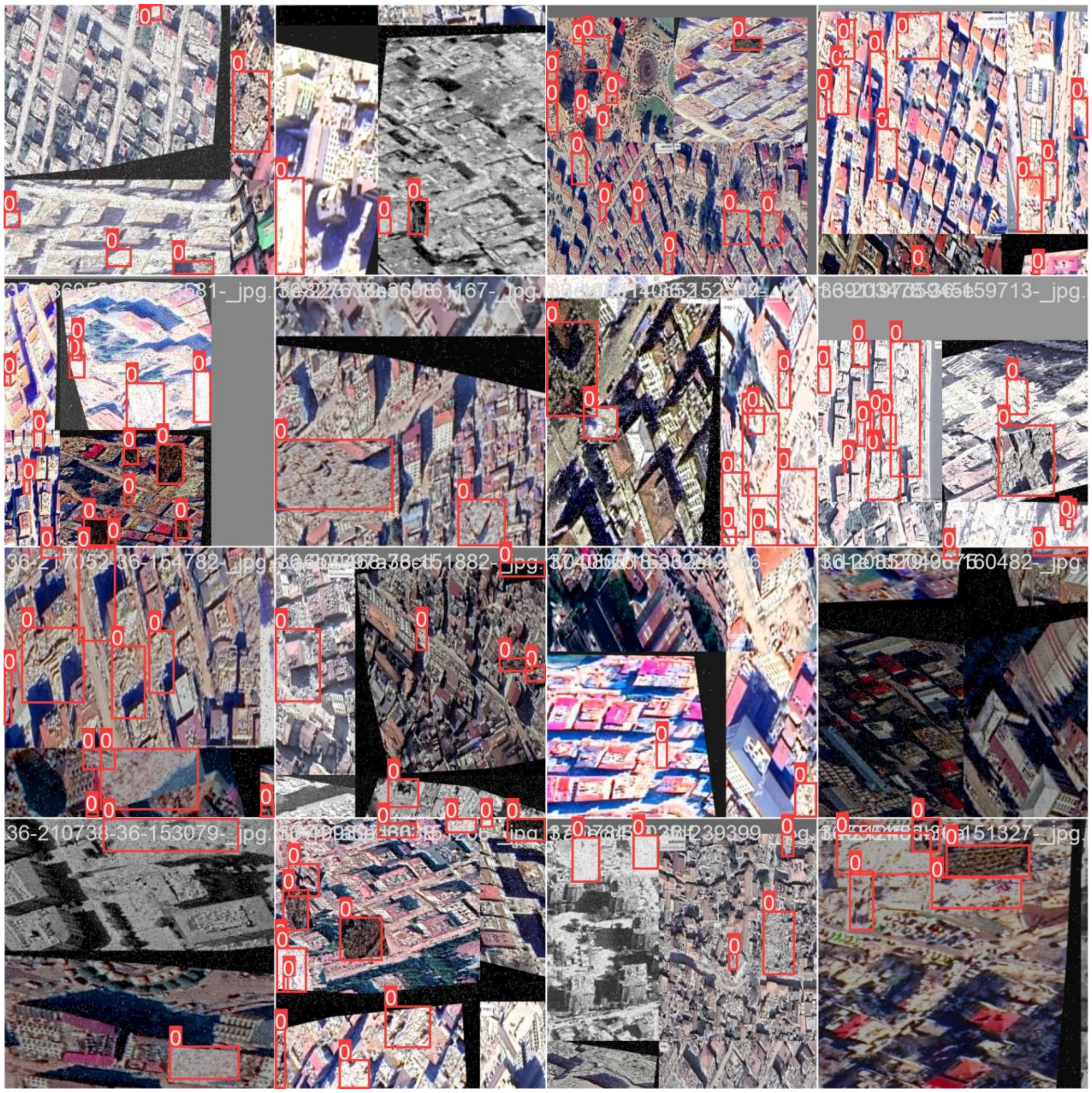


**Figure 8** YOLOv8 object detection train batch number two

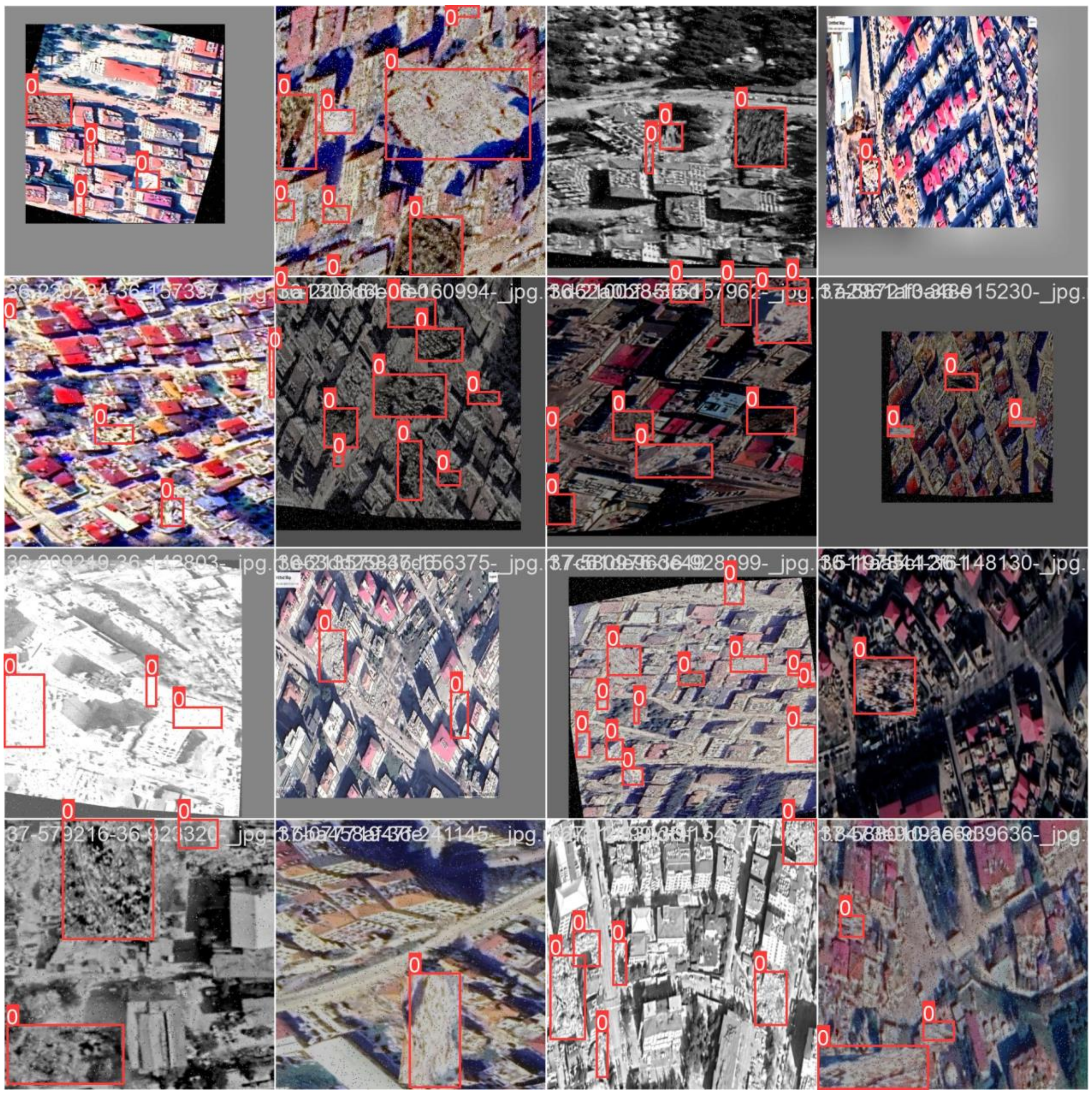


**Figure 9** YOLOv8 object detection train batch number 3332

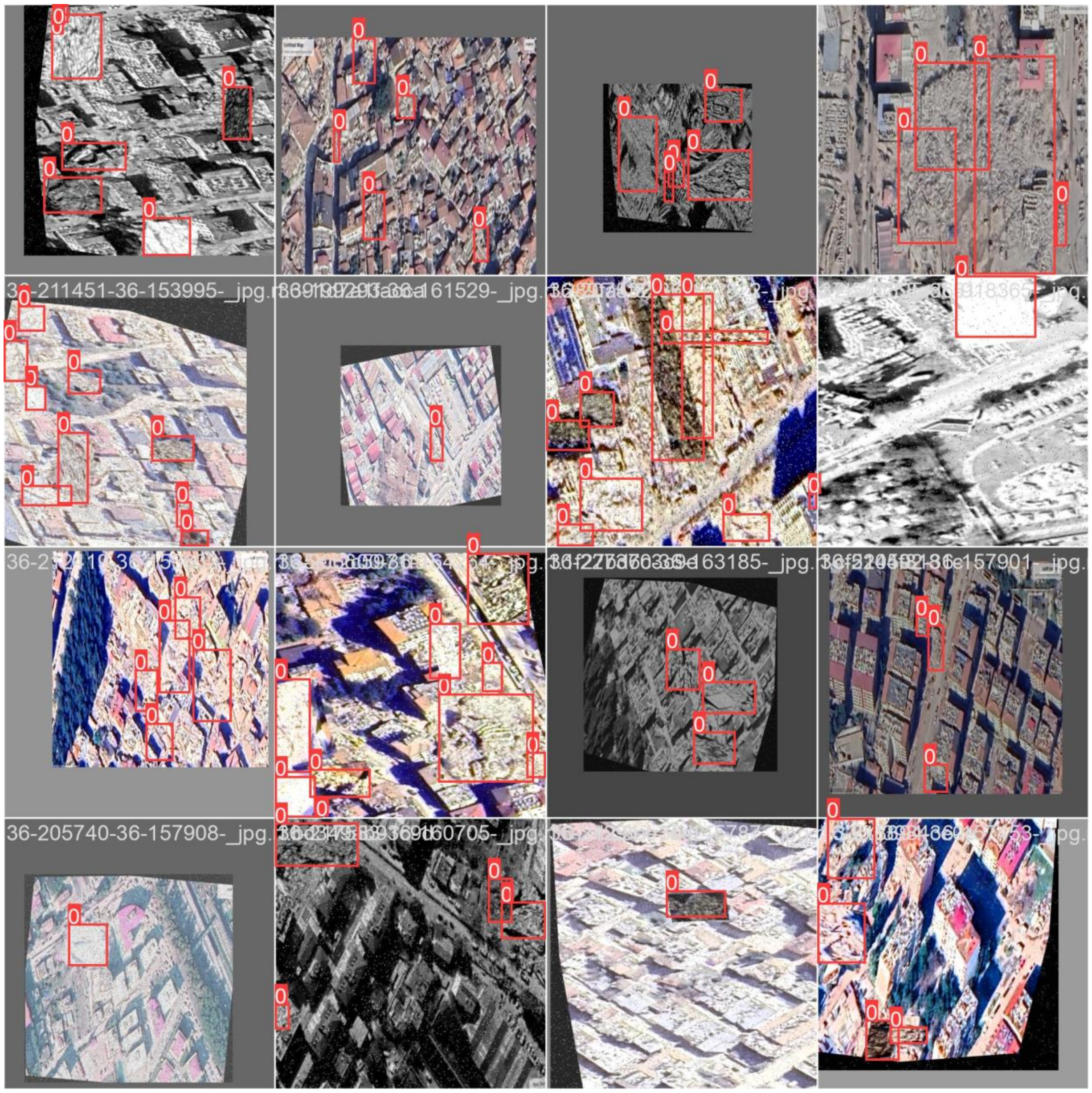


**Figure 10** YOLOv8 object detection train batch number 3333

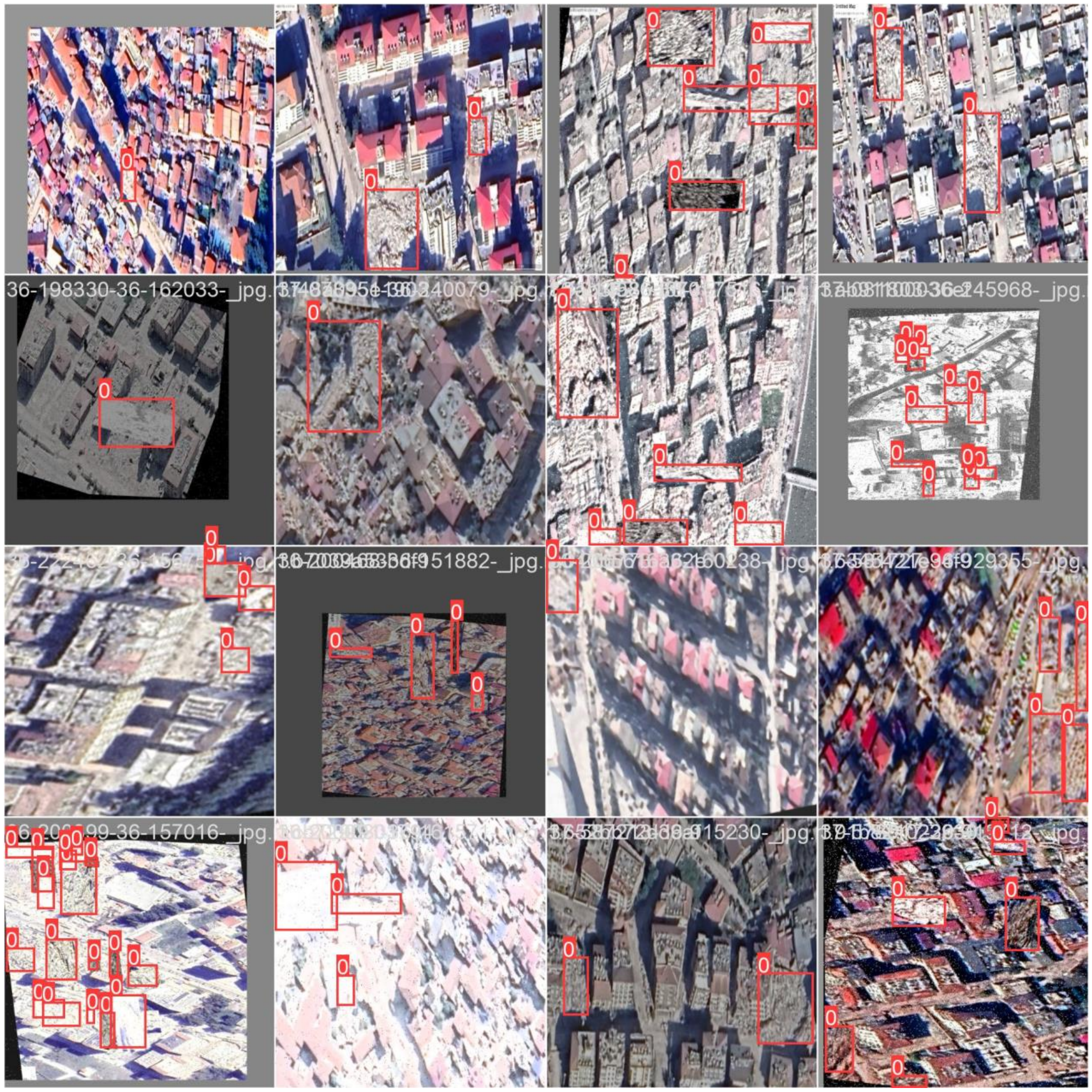


**Figure 11** YOLOv8 object detection train batch number 3334

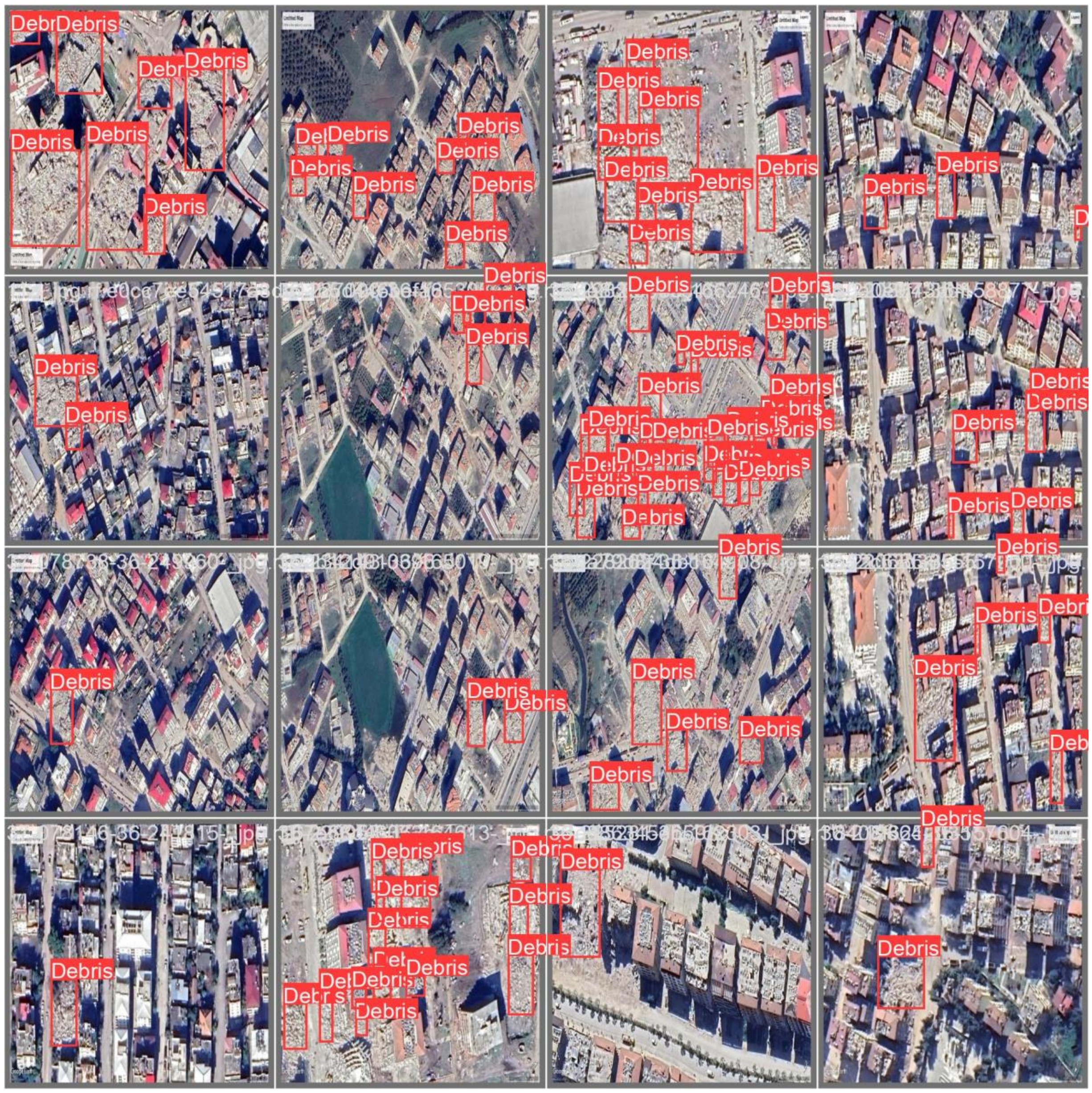


**Figure 12** YOLOv8 object detection validation batch number zero labels

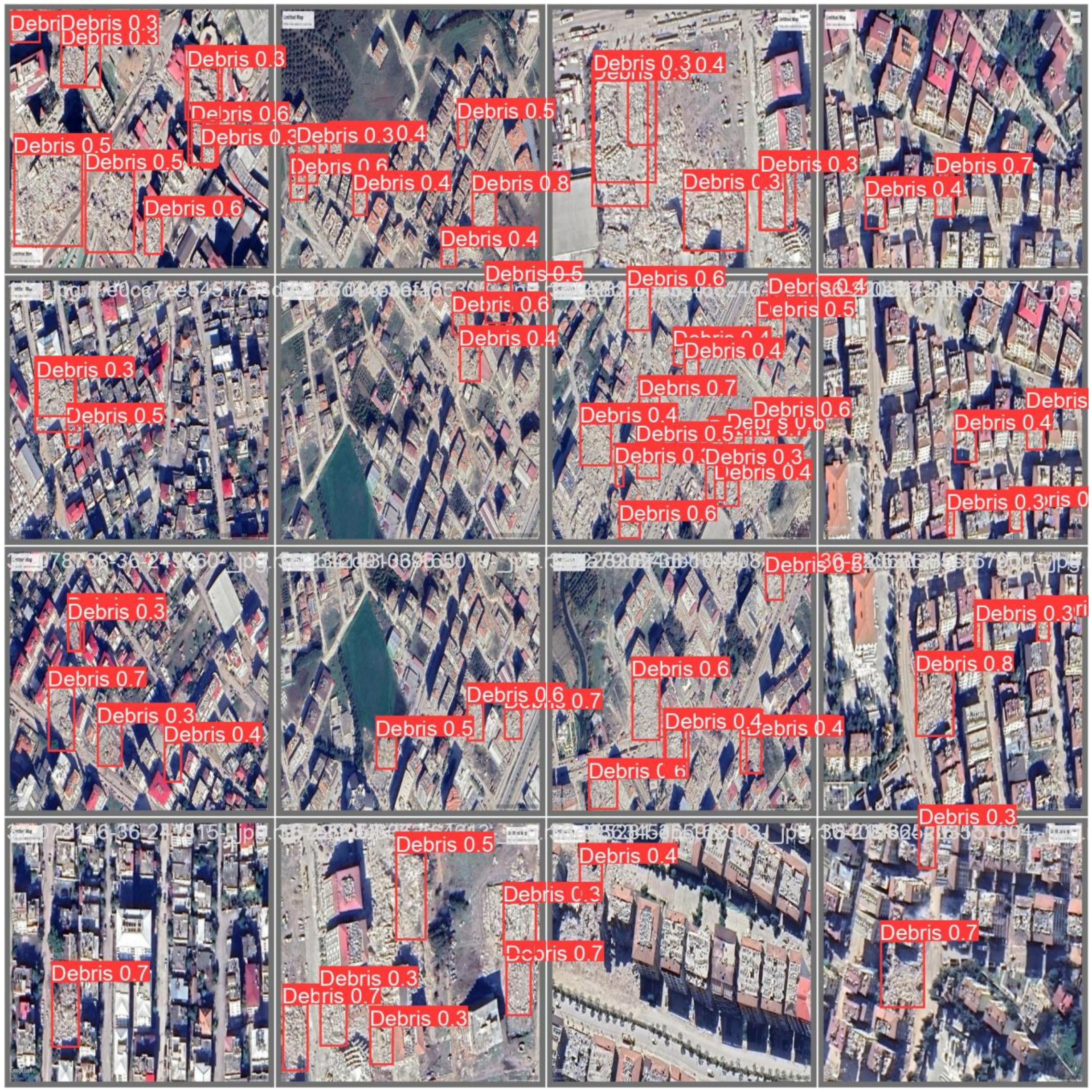


**Figure 13**YOLOv8 object detection validation batch number zero prediction

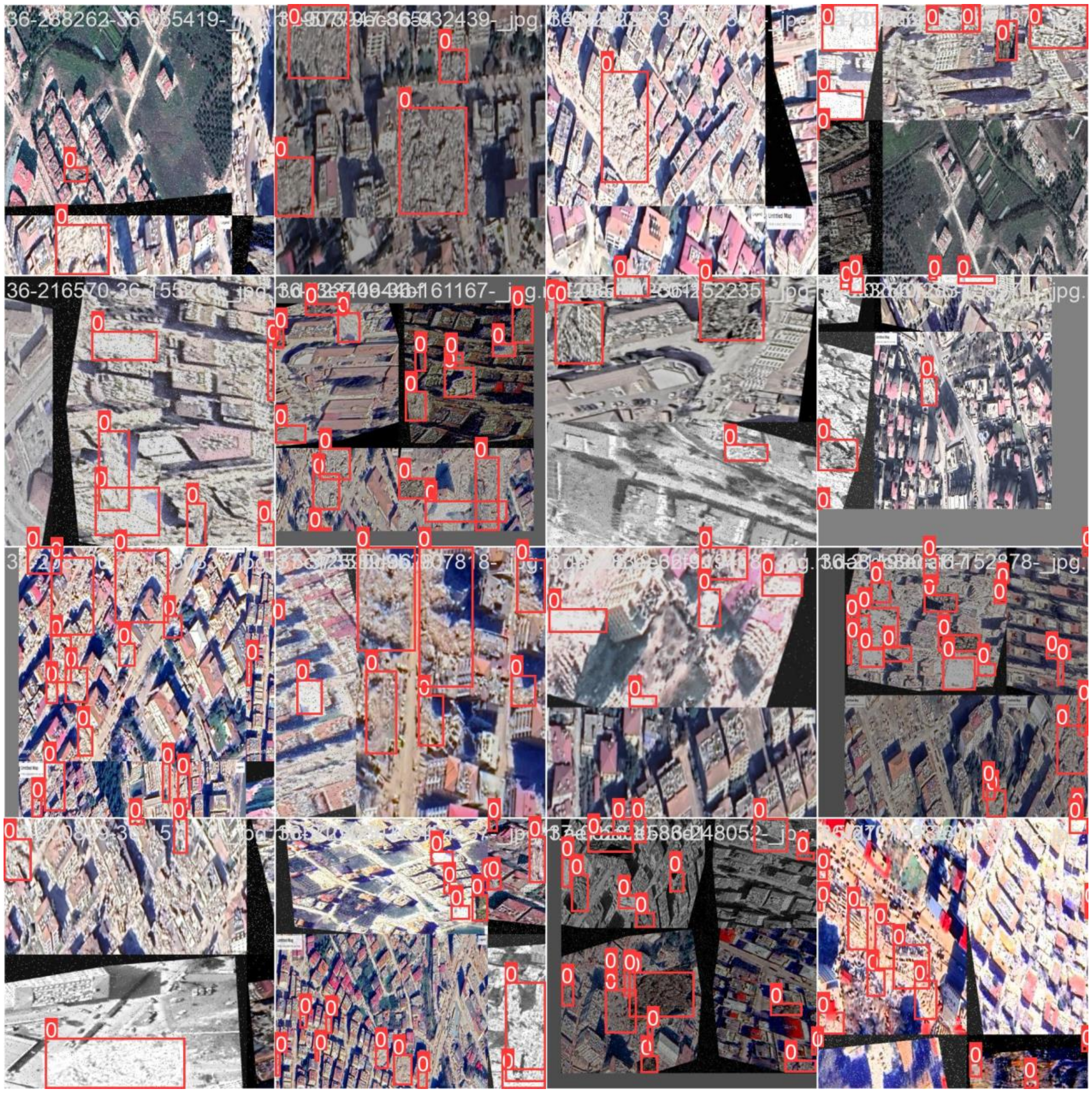


**Figure 14** YOLOv8 object detection validation batch number one labels

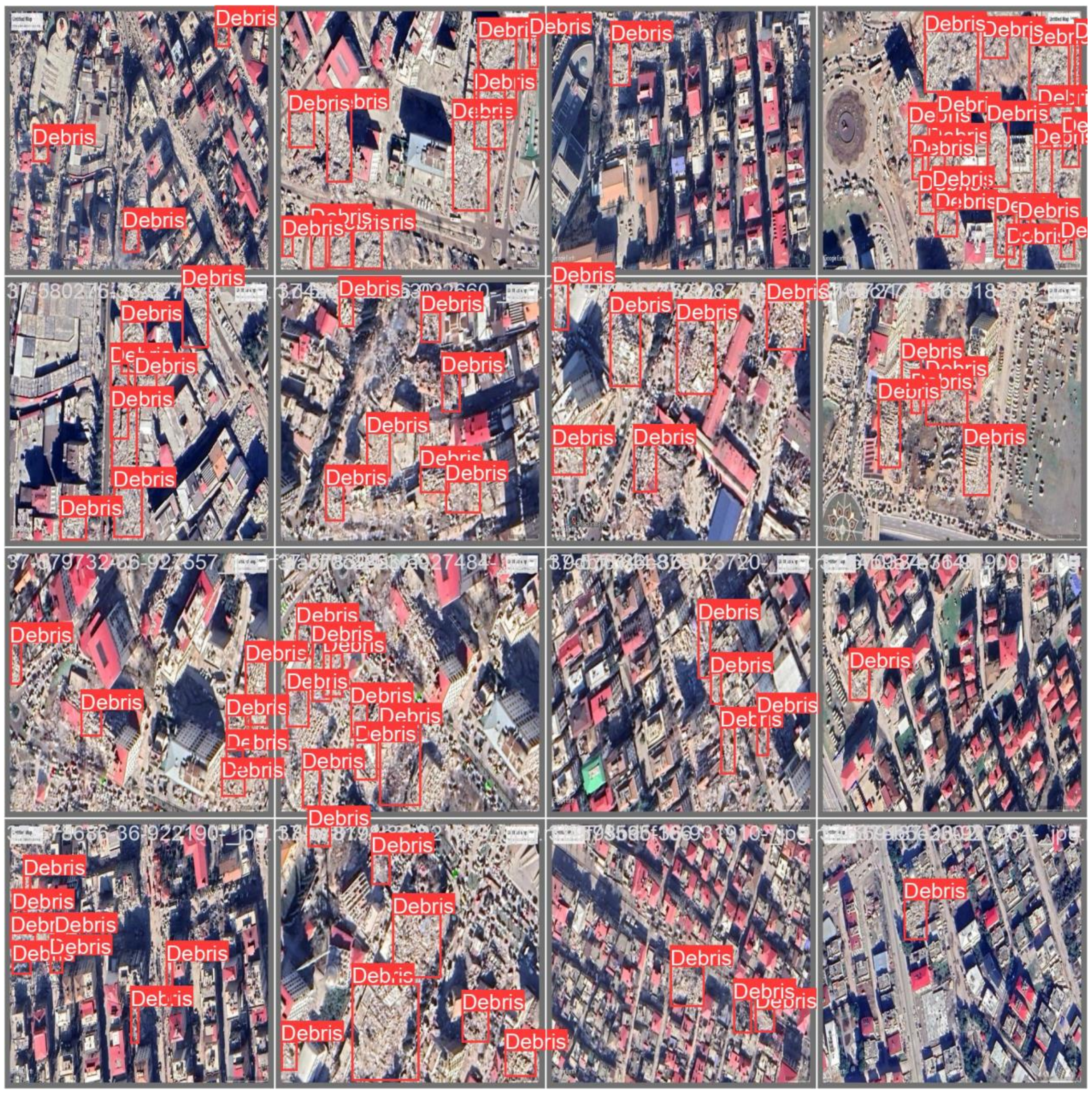


**Figure 15** YOLOv8 object detection validation batch number one labels

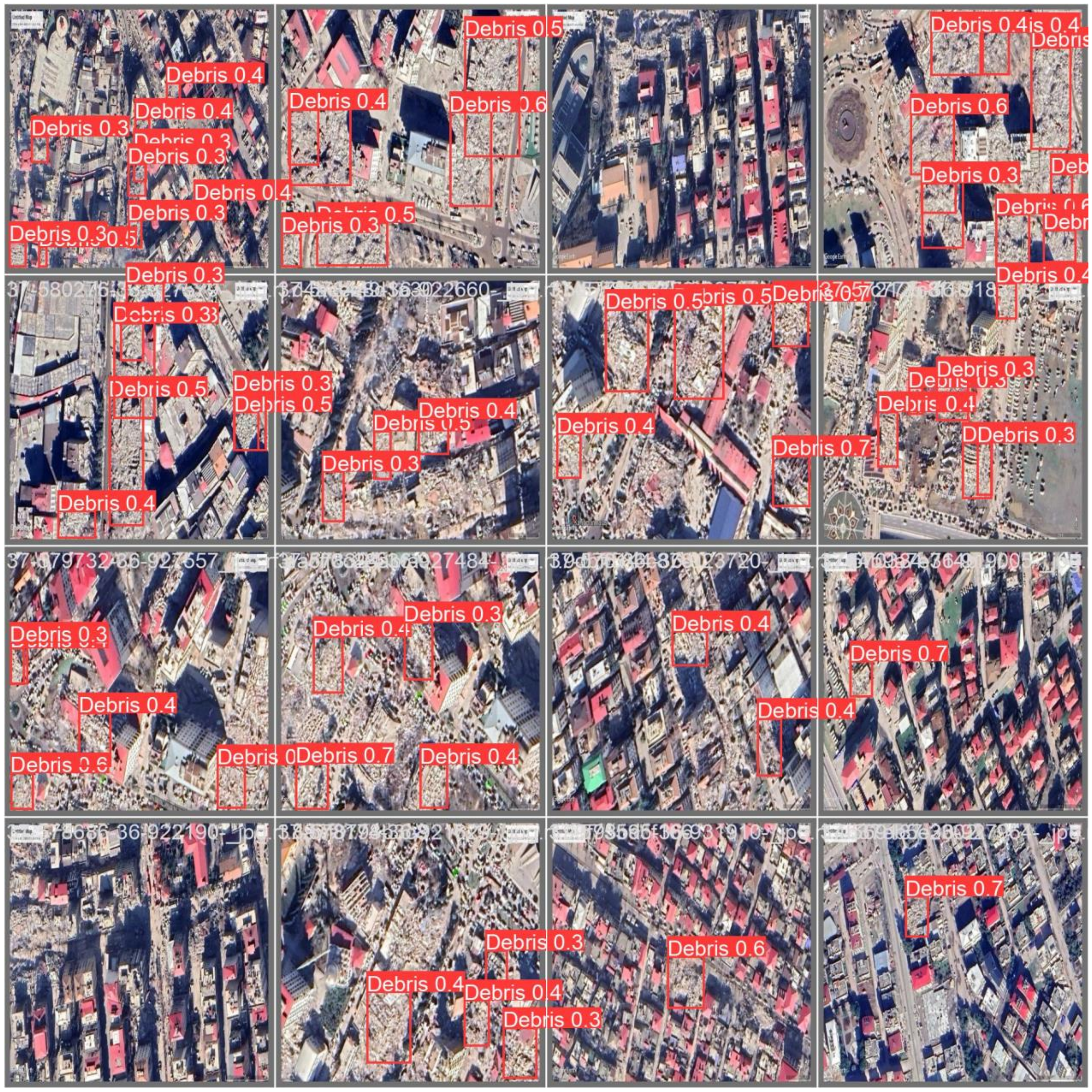


**Figure 16** YOLOv8 object detection validation batch number one prediction

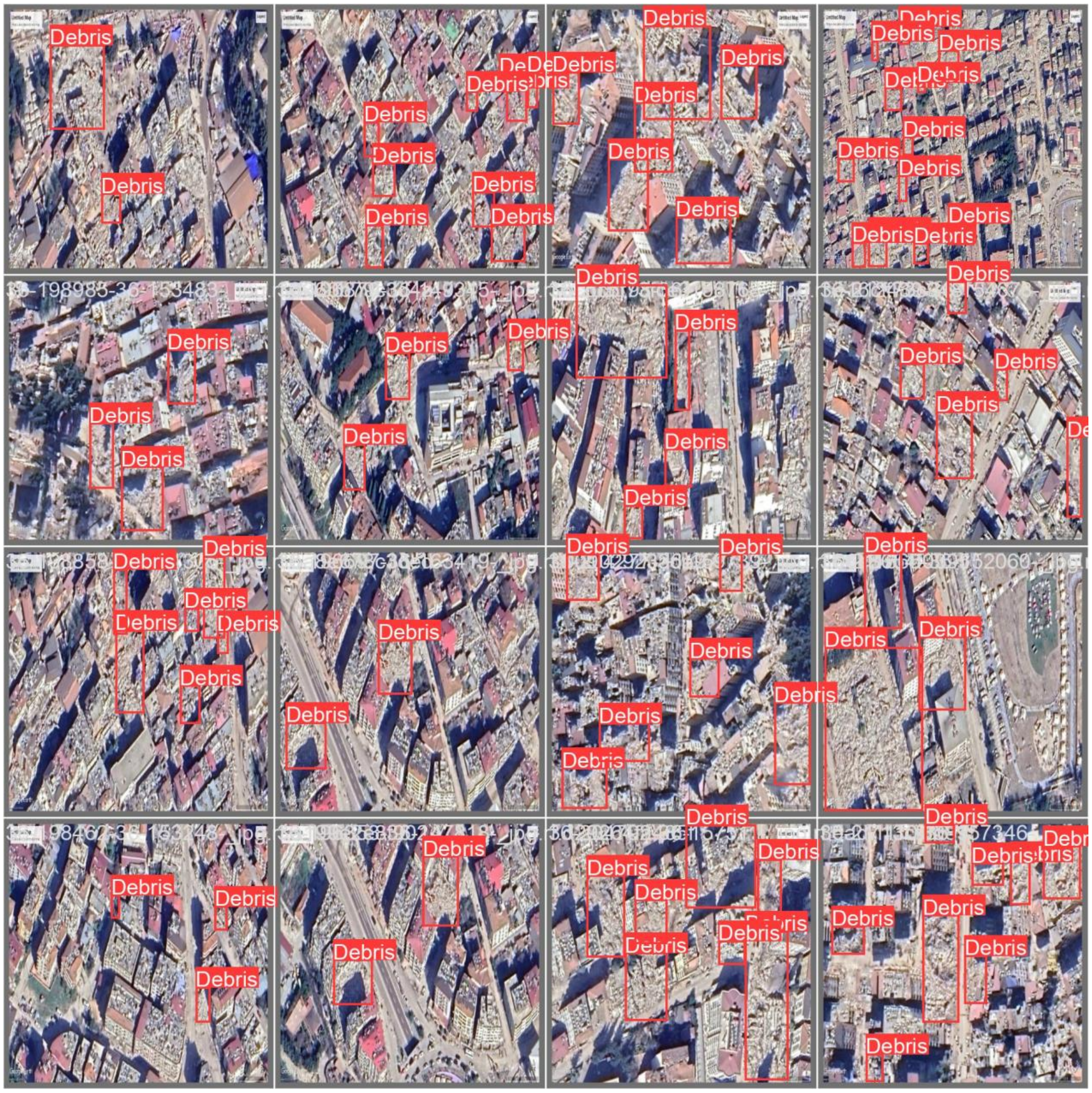


**Figure 17** YOLOv8 object detection validation batch number two labels

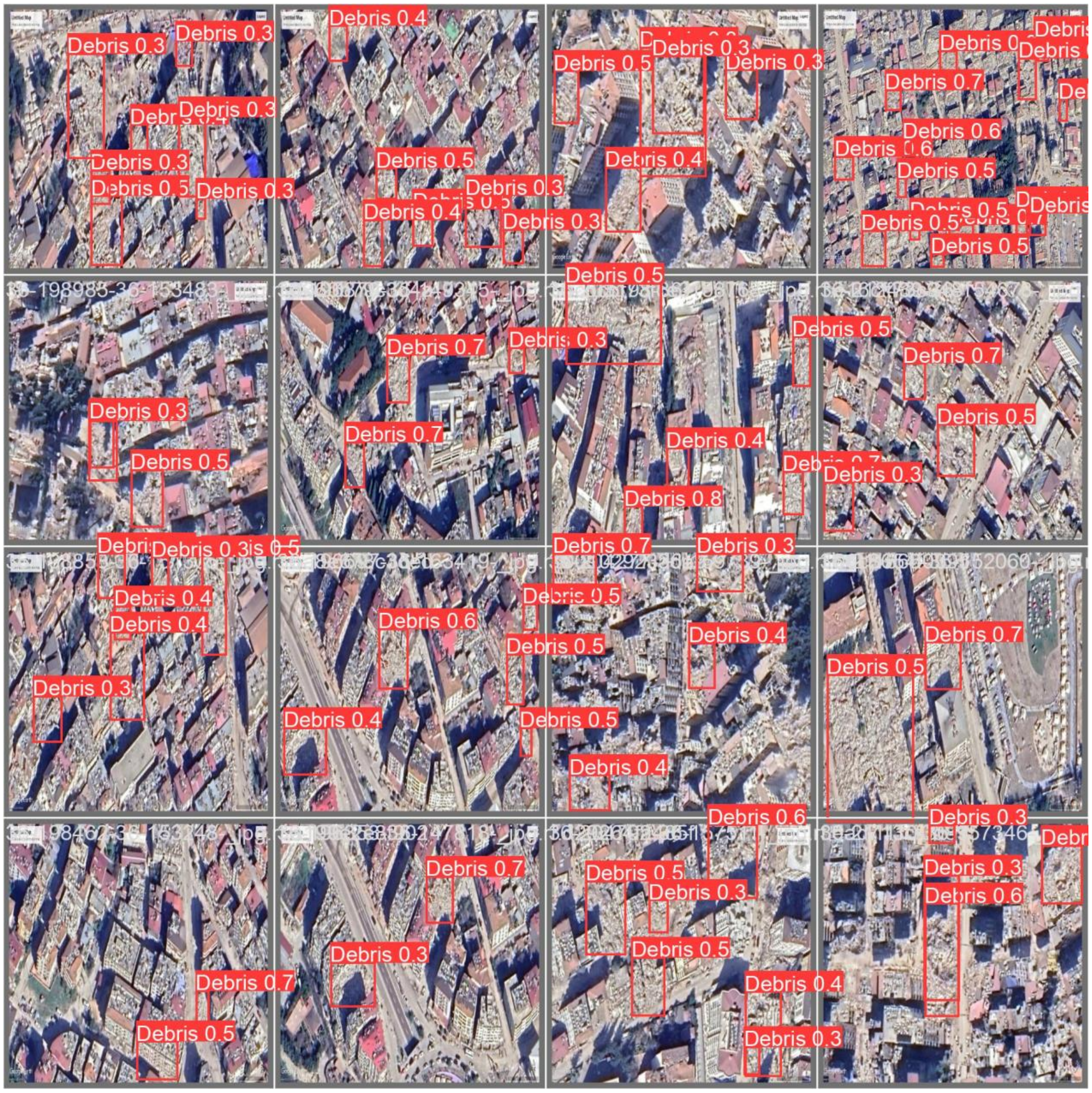


**Figure 18** YOLOv8 object detection validation batch number two prediction

## Appendix 2 – Images from YOLOv8 segmentation Batch and Validation Dataset Results

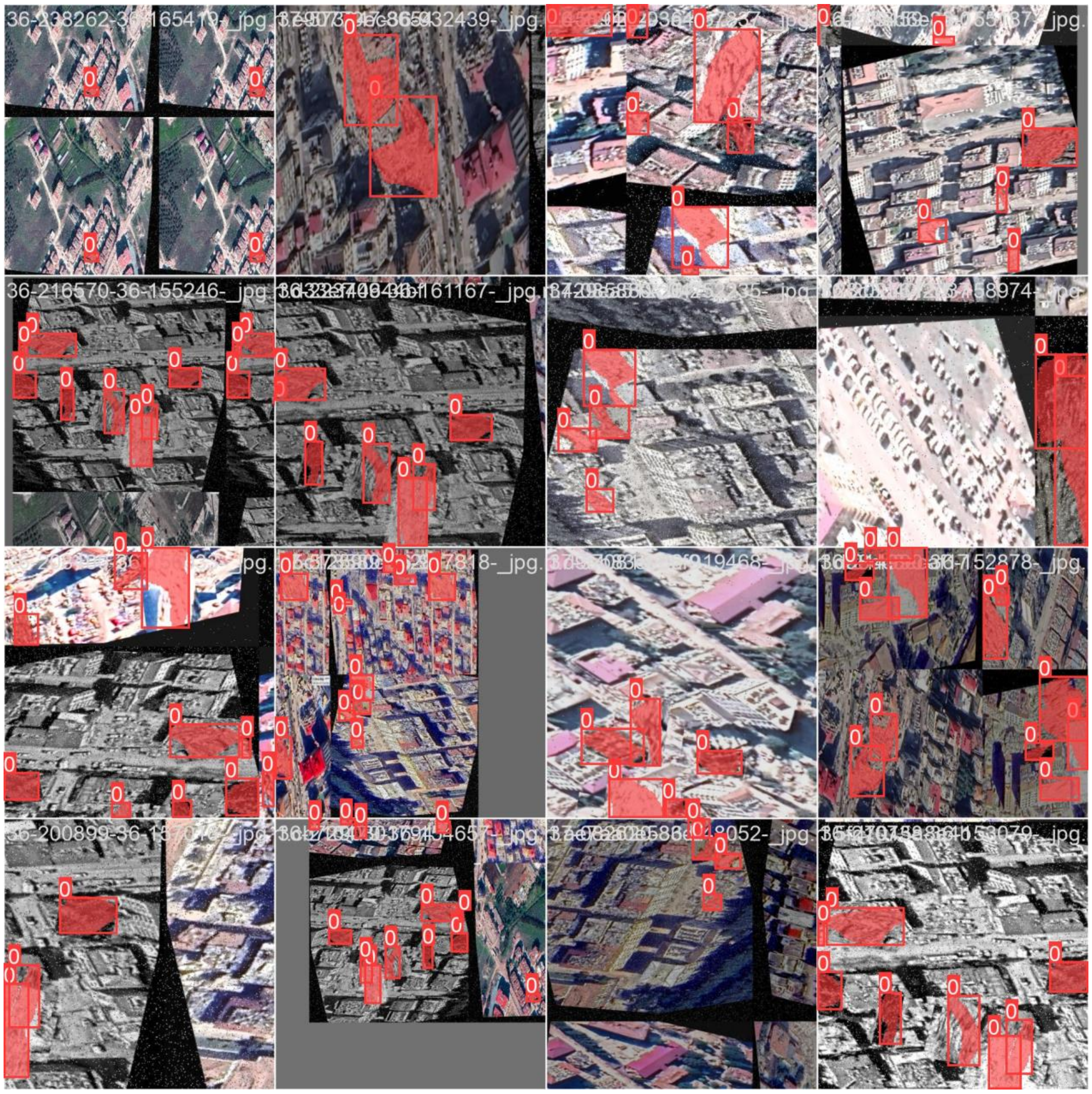


**Figure 1** YOLOv8 segmentation train batch number zero

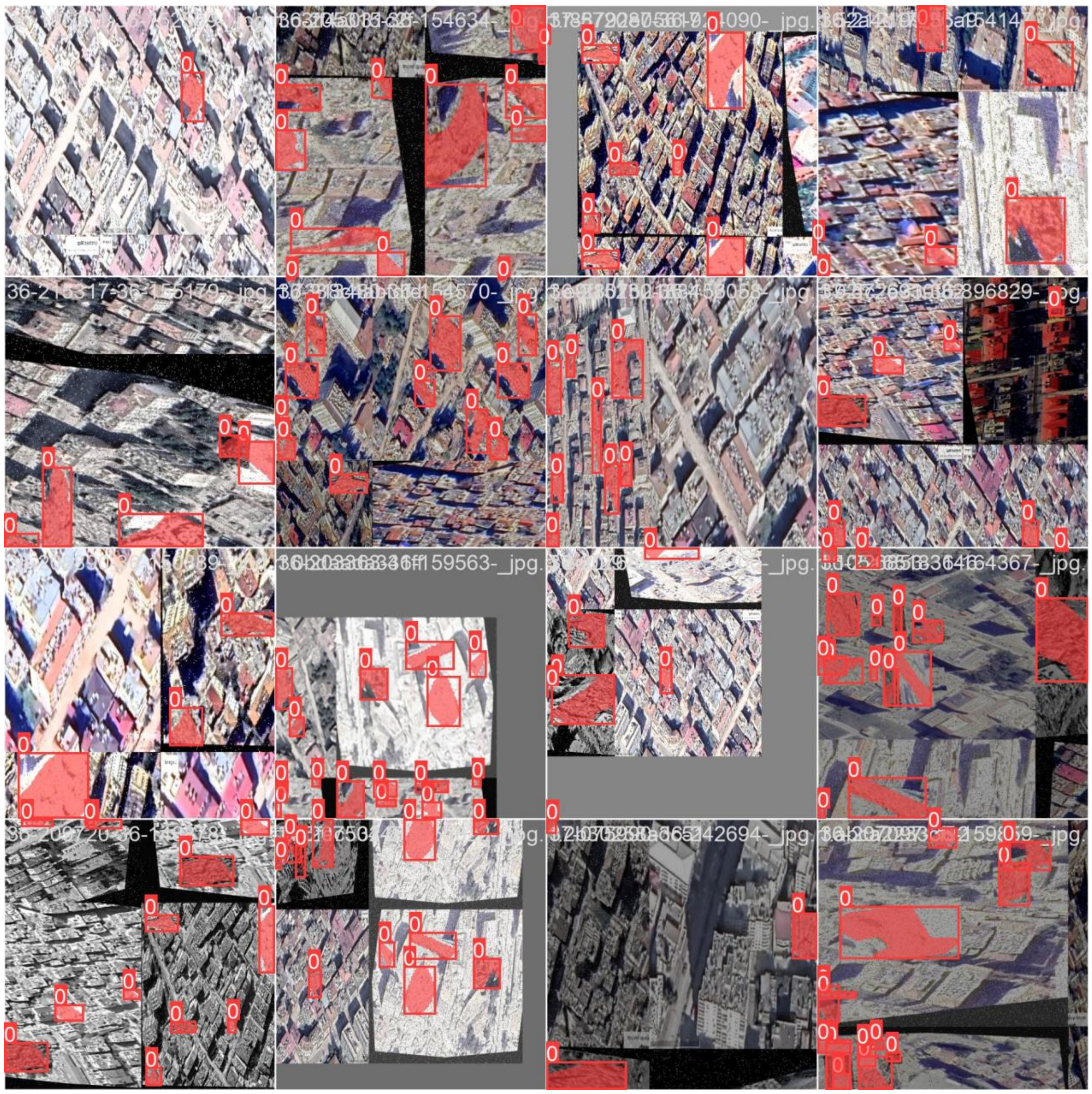


**Figure 2** YOLOv8 segmentation train batch number one

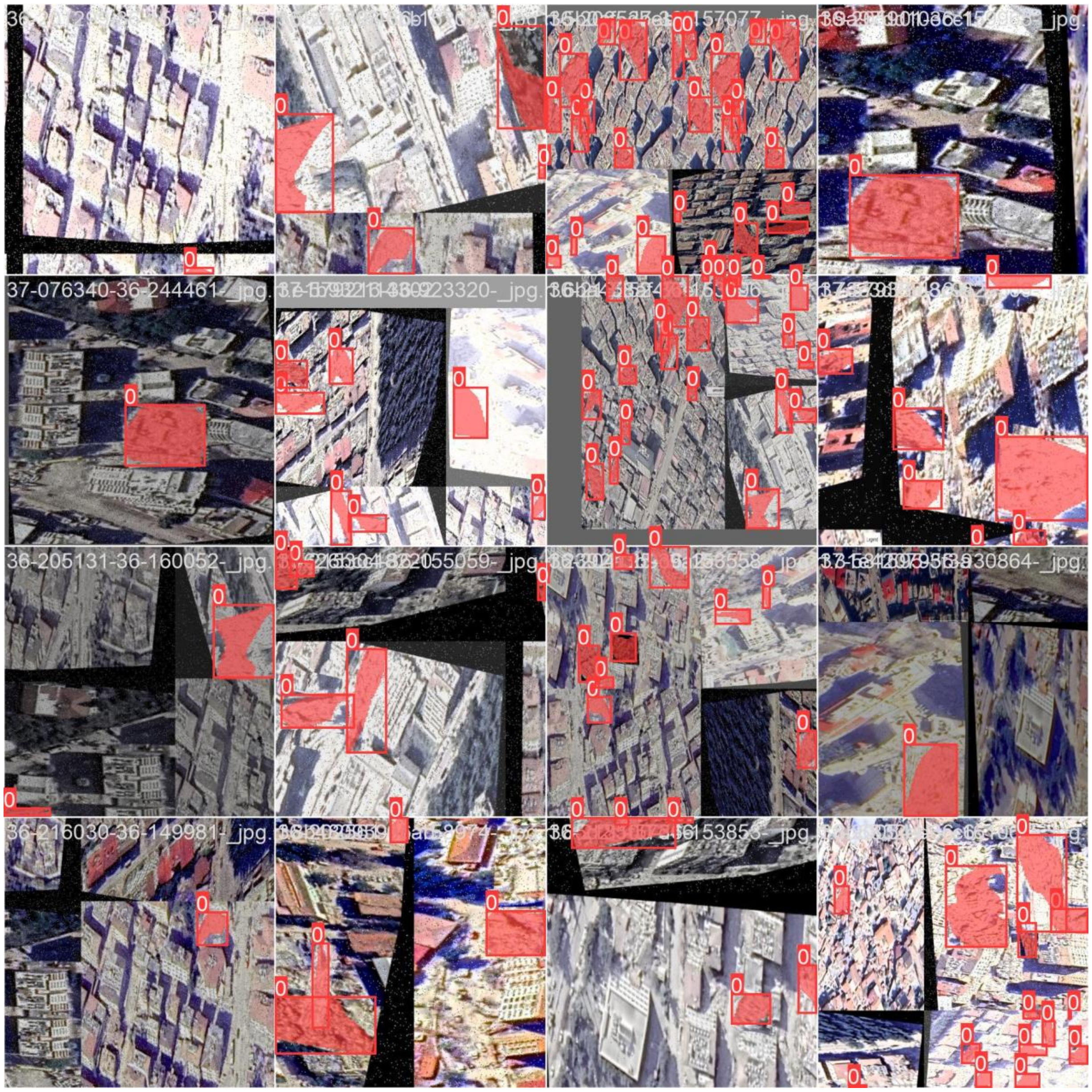


**Figure 3** YOLOv8 segmentation train batch number two

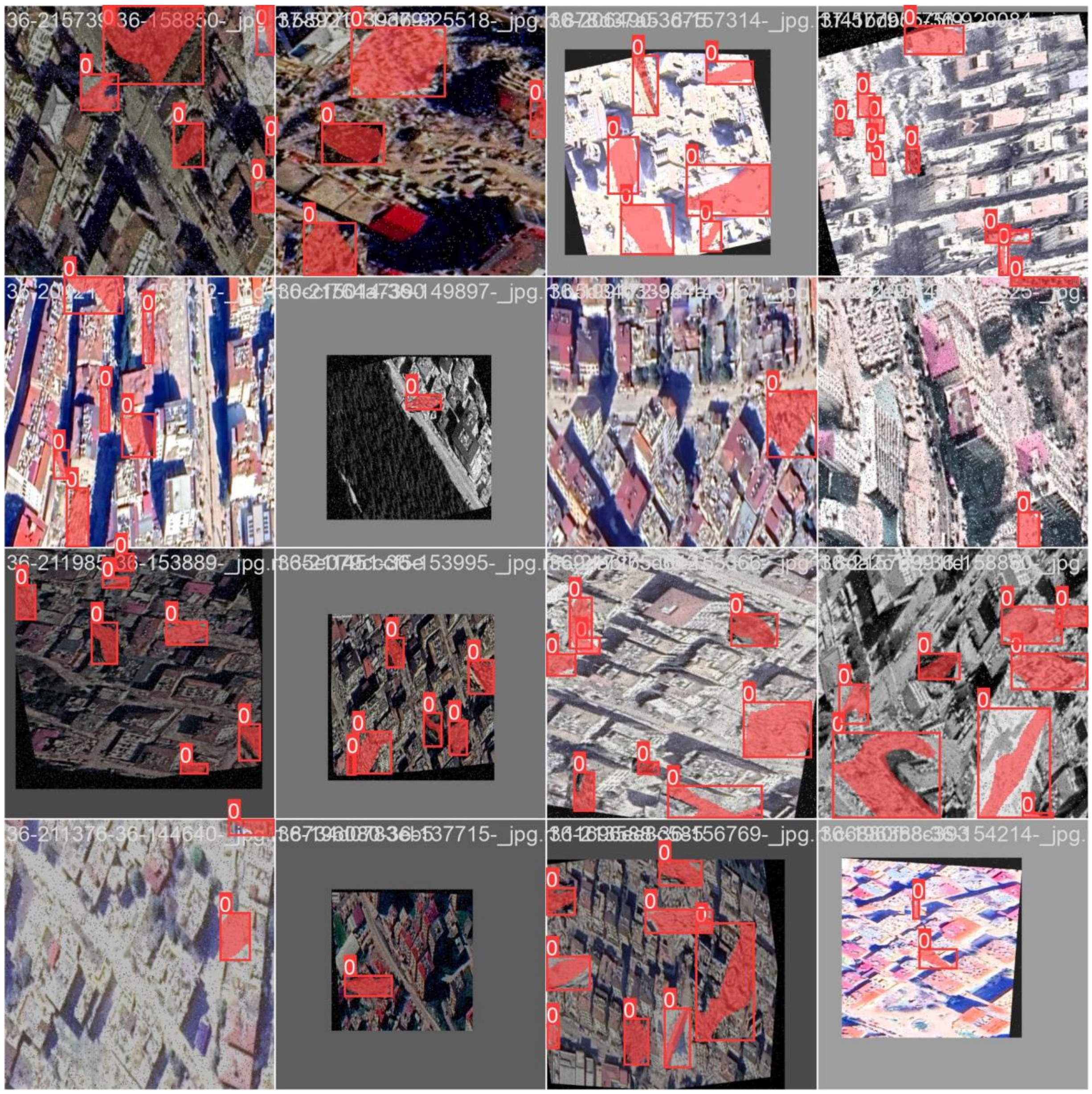


**Figure 4** YOLOv8 segmentation train batch number 3332

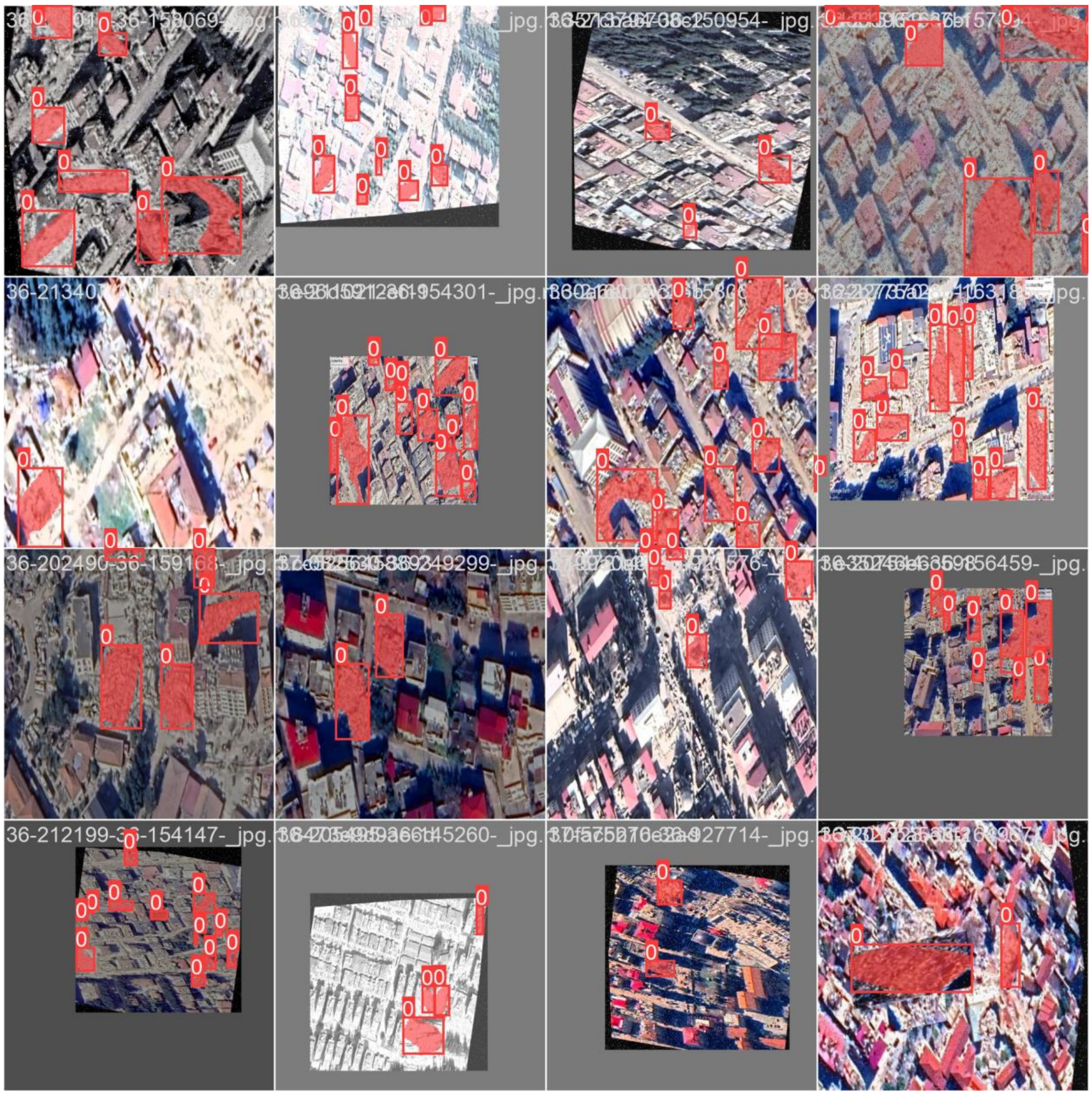


**Figure 5** YOLOv8 segmentation train batch number 3333

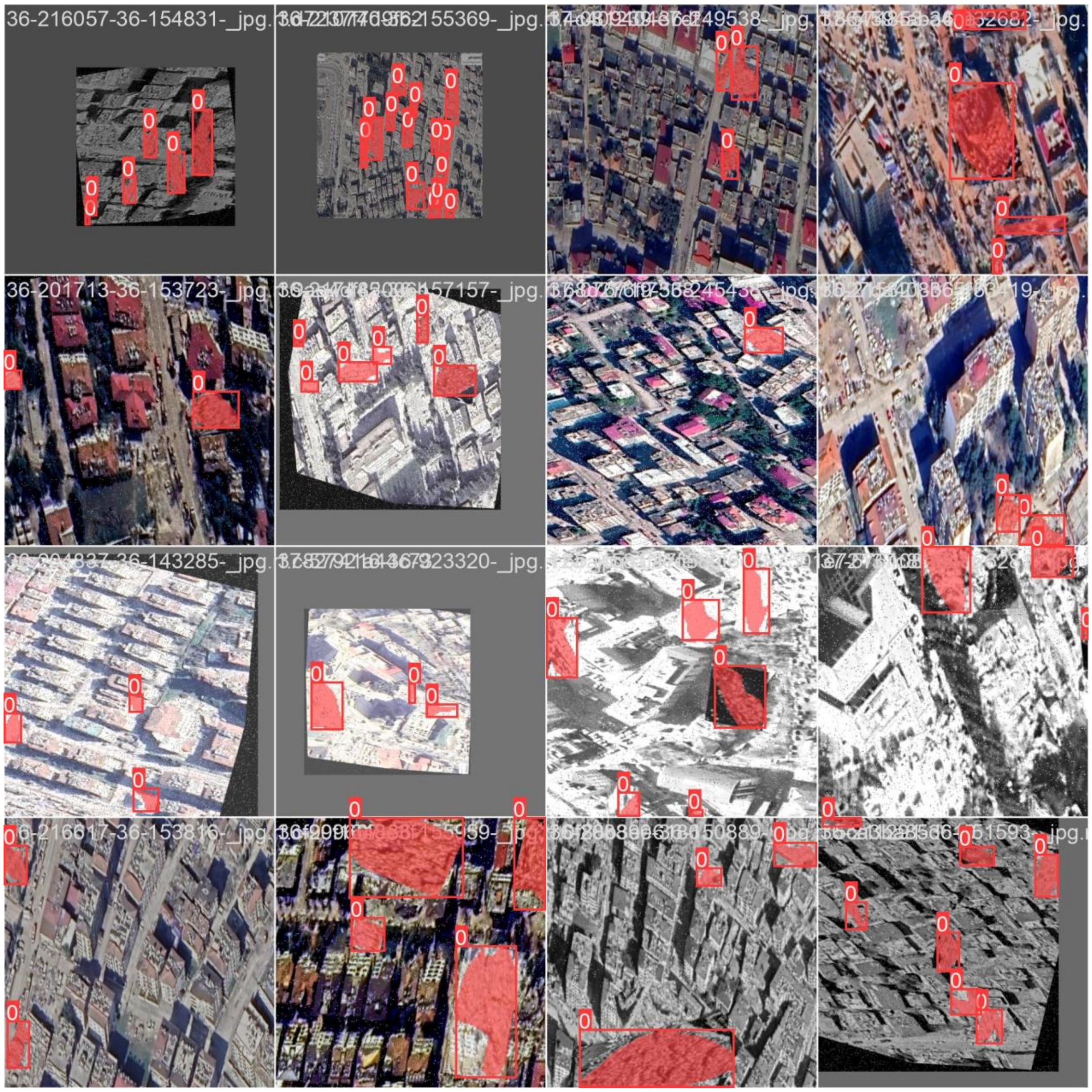


**Figure 6** YOLOv8 segmentation train batch number 3334

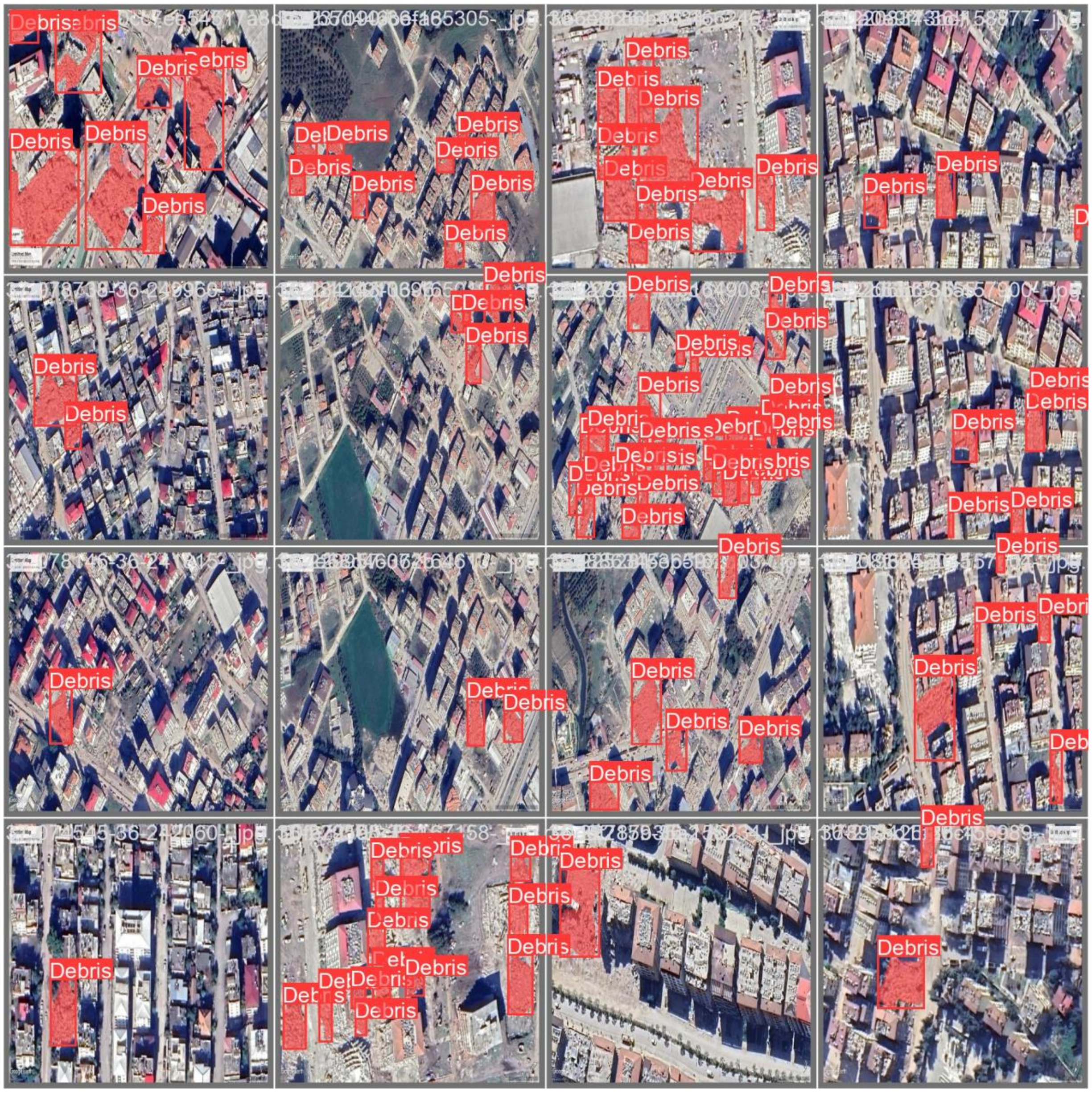


**Figure 7** YOLOv8 segmentation validation batch number zero labels

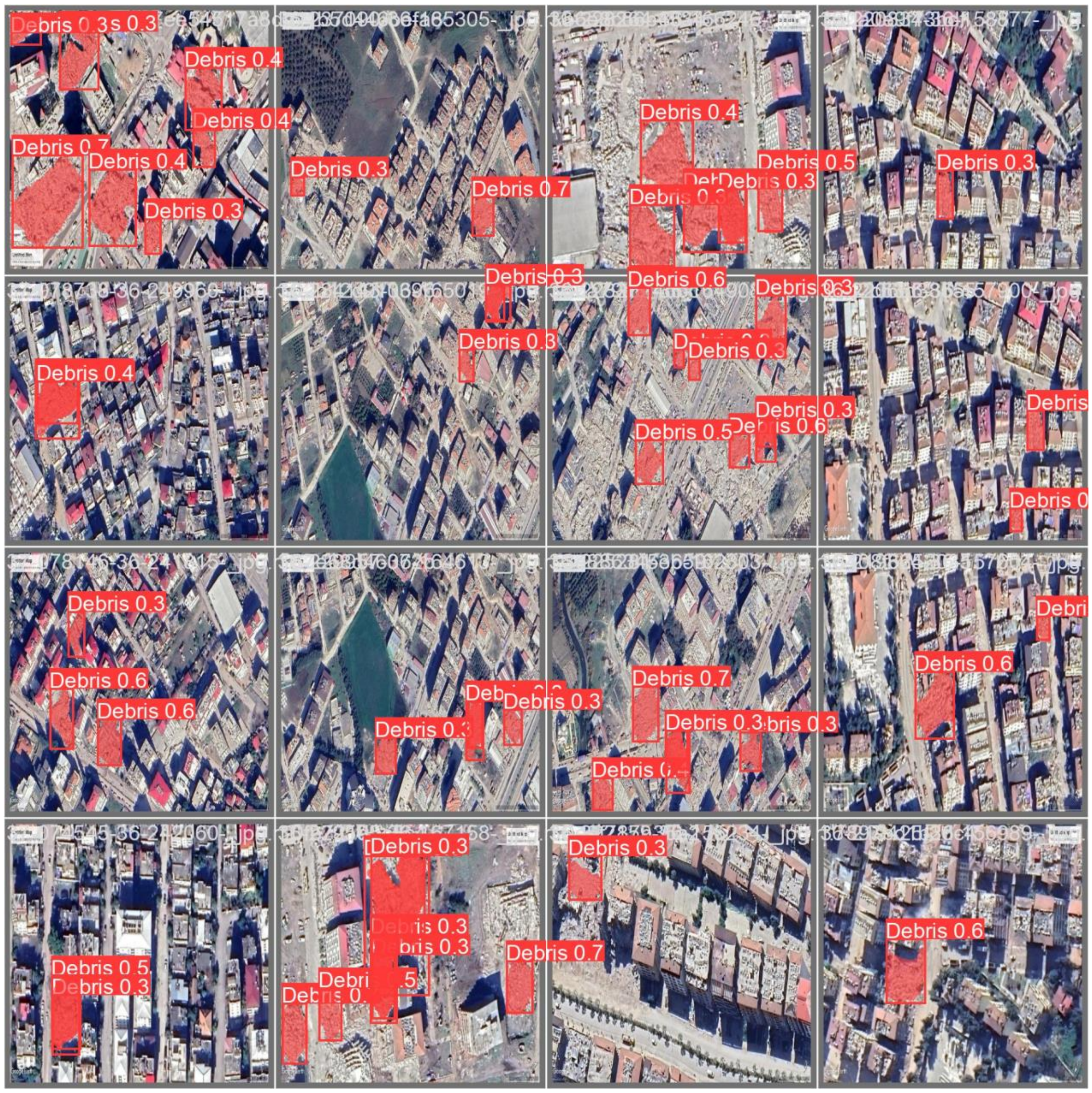


**Figure 8** YOLOv8 segmentation validation batch number zero predictions

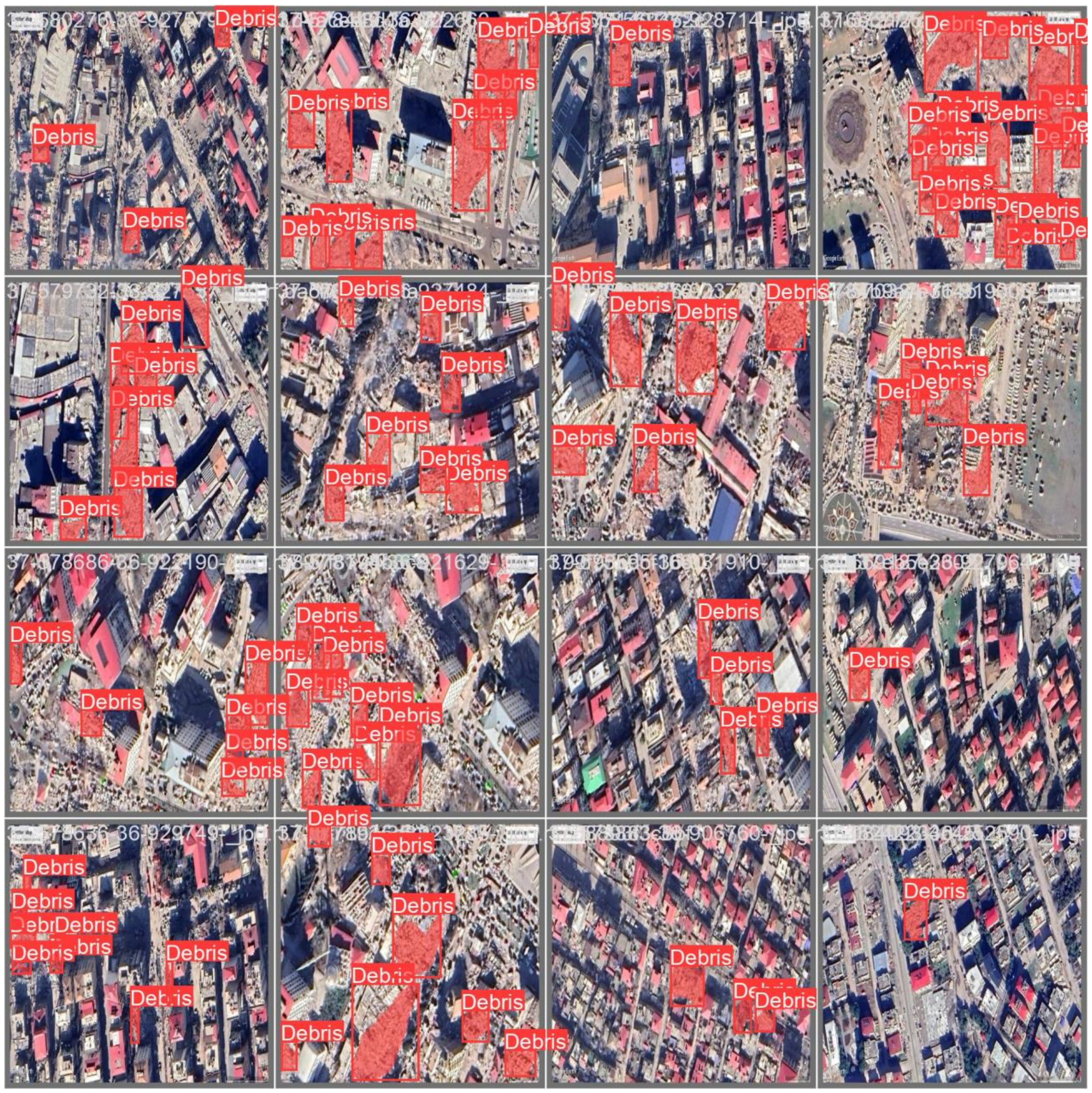


**Figure 9** YOLOv8 segmentation validation batch number one labels

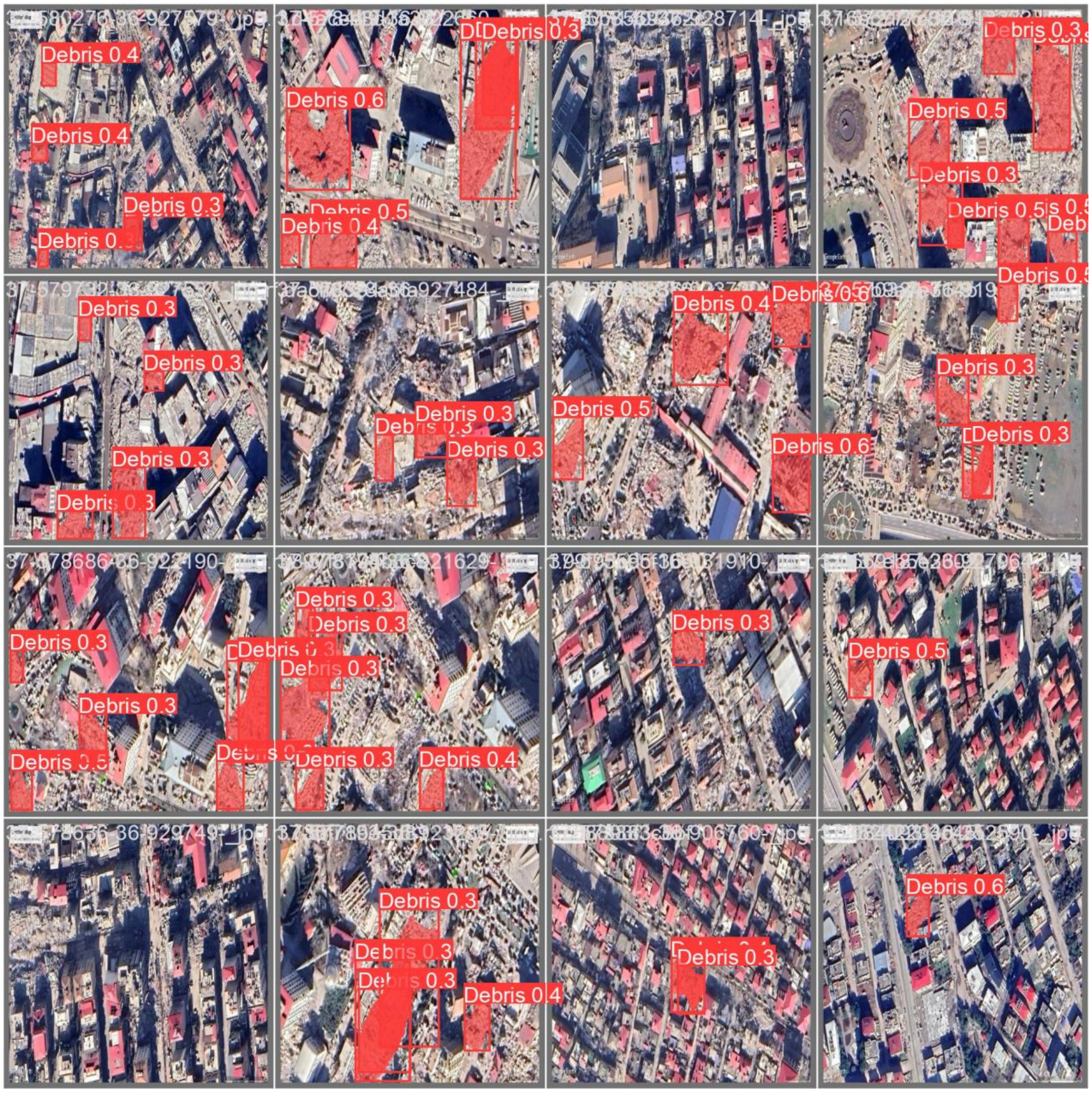


**Figure 10** YOLOv8 segmentation validation batch number one predictions

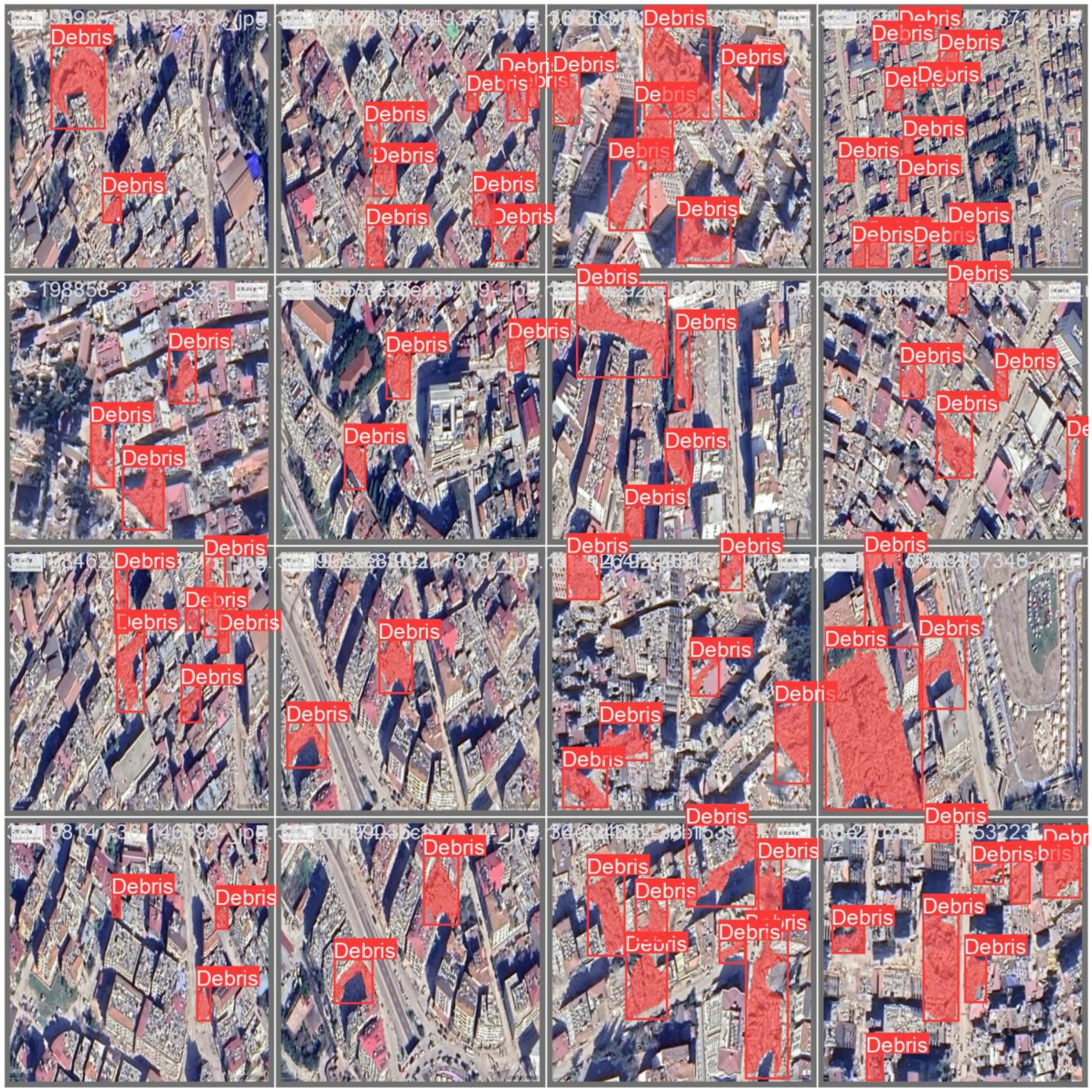


**Figure 11** YOLOv8 segmentation validation batch number two labels

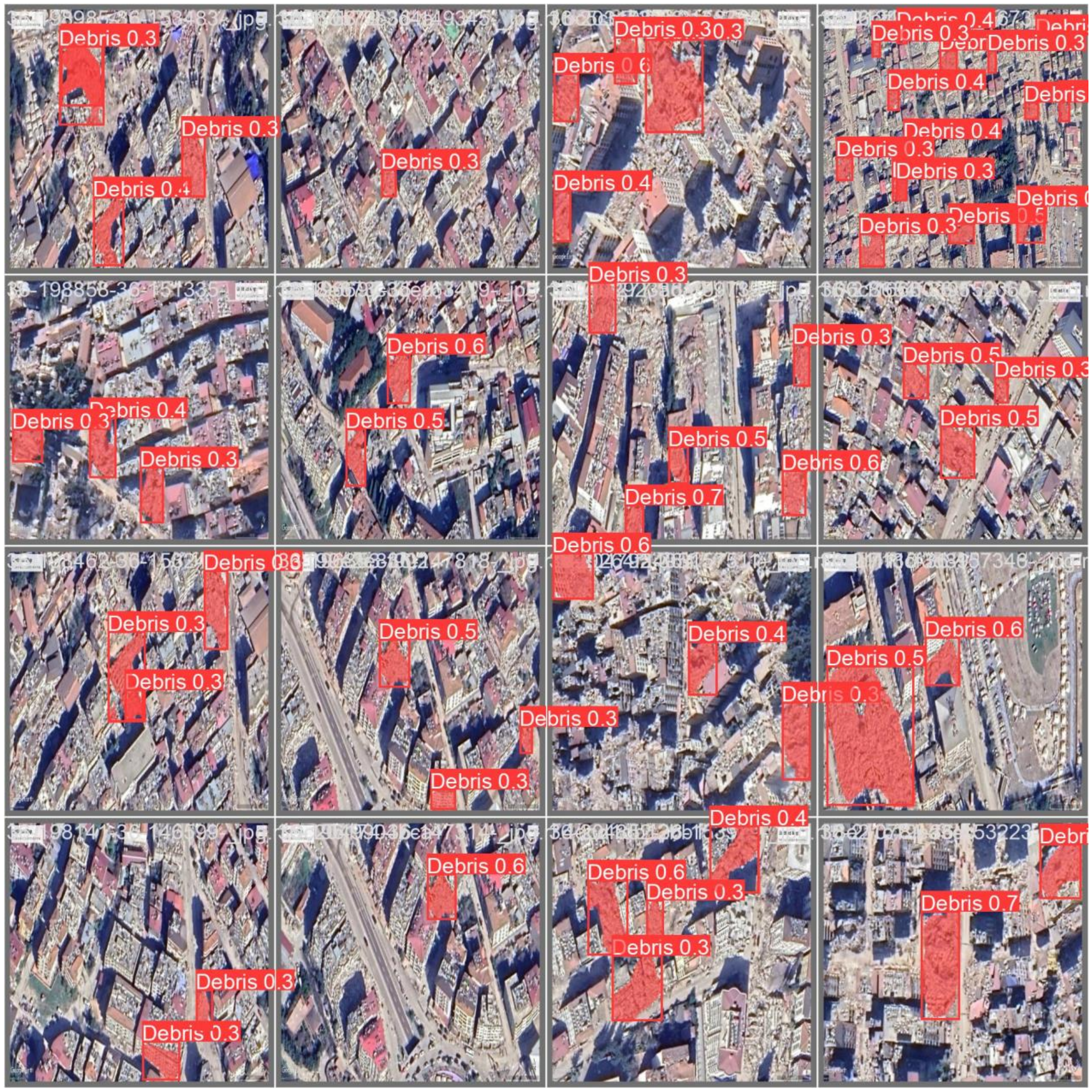


**Figure 12** YOLOv8 segmentation validation batch number two predictions