الجمهورية الجزائرية الديمقراطية الشعبية
People's Democratic Republic of Algeria
وزارة التعليم العالي والبحث العلمي
Ministry of Higher Education and Scientific Research

Nº Ref :……

University Center
Abd Elhafid Boussouf Mila

Mathematics & Computer Science Institute
Computer Science Department

Thesis prepared for the Master's degree

Specialty: Artificial Intelligence and its Applications (I2A)

DisasterVision: a computer vision system for disaster management

Prepared by:

- **KERKATOU Raid Anis**
- **SELLAHI Abdelkadir**

Jury Reviewed :

Dr. Bencheikh Lehocine Madjed	**……**	**President**
Dr. Hadji Athmane	**……**	**Examiner**
Dr. Boulmerka Aissa	**……**	**Supervisor**

Academic Term: 2023/2024

Acknowledgement

First of All, Thanks to Allah for giving us the courage, health, power, and patience to complete this work.

We deeply appreciate our supervisor, ***Dr. Aissa Boulmerka****, for his invaluable guidance, support, and availability throughout this project. His expert advice and assistance were crucial in helping us understand the objective.*

Last but not least, Our families deserve our deep appreciation so we want to express it for their unwavering support and endless encouragement throughout our studies. Their love, patience, and sacrifices have been a constant source of motivation during challenging times, and we could not have achieved this without their support.

Thank you to everyone who has been a part of this long journey. Your contributions, no matter how small, have been deeply appreciated and have significantly impacted our work.

KERKATOU RAID ANIS * SELLAHI ABDELKADIR**

Abstract

By using drone images, this thesis investigates how computer vision methods can effectively assess damage caused by natural disasters, such as floods. It uses deep learning techniques, particularly Artificial Neural Networks, which have been extensively explored for tasks such as Semantic Segmentation, Object Detection, and Visual Question Answering. The primary objective of this research is to identify and classify various damages resulting from disasters. The study evaluates several architectures of Artificial Neural Networks, including Deeplabv3 and Unet + Resnet backbone using the Floodnet dataset in the context of semantic segmentation. As for the detection and classification of different types of damage, object detection techniques like YOLOv8n are applied using the mentioned dataset above. Lastly for visual questions answering the process of answering and prediction go through BERT and GPT-2.

key-words: deep learning, Convolutional neural networks, Image Classification, Semantic segmentation, Object detection, Unet, DeeplabV3+, Resnet, BERT , GPT-2, YOLOv8n, Visual Question Answering.

ملخص

باستخدام صور الطائرات بدون طيار، تبحث هذه الأطروحة في كيفية استخدام طرق الرؤية الحاسوبية لتقييم الأضرار الناجمة عن الكوارث الطبيعية، مثل الفيضانات، بفعالية عالية حيث تستخدم هذه الدراسة تقنيات التعلم العميق، ولا سيما الشبكات العصبية ، التي تم استغلالها بشكل مكثف لمهام مثل التقسيم الدلالي، واكتشاف الأجسام، والإجابة على الأسئلة المرئية. الهدف الرئيسي من هذا البحث هو تحديد وتصنيف الأضرار المختلفة الناتجة عن الكوارث.حيث يتم تقييم العديد من هياكل الشبكات العصبية ، بما في ذلك Deeplabv3+ و Unet مع ResNet باستخدام مجموعة صور Floodnet في سياق التقسيم الدلالي. بالنسبة لاكتشاف وتصنيف أنواع الأضرار المختلفة، يتم تطبيق تقنيات اكتشاف الأجسام مثل YOLOv8n باستخدام مجموعة الصور المذكورة أعلاه. وأخيراً، للإجابة على الأسئلة المرئية، يتم تنفيذ عملية الاجابة عن الاسئلة باستخدام تقنيتي Bert و GPT-2 .

الكلمات المفتاحية: التعلم العميق، الشبكات العصبية ، تصنيف الصور، التقسيم الدلالي، اكتشاف الأجسام، Unet، Deeplabv3+، ResNet، BERT، GPT-2، YOLOv8n، الإجابة على الأسئلة المرئية.

Résumé

En utilisant des images de drones, cette projet examine comment les méthodes de vision par ordinateur peuvent évaluer efficacement les dommages causés par des catastrophes naturelles, telles que les inondations. Elle utilise des techniques d'apprentissage profond, en particulier les réseaux de neurones artificiels, qui ont été largement explorés pour des tâches telles que la segmentation sémantique, la détection d'objets et les réponses visuelles aux questions. L'objectif principal de cette recherche est d'identifier et de classifier divers dommages résultant de catastrophes. L'étude évalue plusieurs architectures de réseaux de neurones artificiels, y compris Deeplabv3+ et Unet avec backbone Resnet, en utilisant le jeu de données Floodnet dans le contexte de la segmentation sémantique. En ce qui concerne la détection et la classification des différents types de dommages, des techniques de détection d'objets comme YOLOv8n sont appliquées en utilisant le jeu de données mentionné ci-dessus. Enfin, pour les réponses visuelles aux questions, le processus de repondre et la prédiction passent par BERT et GPT-2.

Mots-clés : Apprentissage Profond, Réseaux de Neurones Convolutifs, Classification d'images, Segmentation Sémantique, Détection d'Objets, Unet, DeeplabV3+, ResNet, BERT, GPT-2, YOLOv8n, Réponses Visuelles aux Questions.

Contents

List of Figures

List of Tables

List of Abbreviations

DL	Deep Learning
NN	Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ED	Encoder-Decoder
GUI	Graphical User Interface
IDE	Integrated Development Environment
CPU	Central Processing Unit
GPU	Graphics processing Unit
IoU	Intersection-Over-Union
LR	Learning Rate
FC	Fully Connected
ResNet	Residual Neural Network
NLP	Natural Language Processing
API	Application Programming Interface
FCN	Fully Convolutional Network
ViLT	Vision and Language Transformer

General introduction

In the field of disaster management, the ability to swiftly assess the damage and respond effectively is paramount. Unmanned Aerial Vehicles (UAVs), commonly known as drones, have emerged as indispensable tools in addressing these challenges. Leveraging their capabilities, drones facilitate the acquisition of high-resolution imagery and essential data from areas that are otherwise inaccessible or perilous.

The integration of deep learning methodologies such as convolutional and recurrent Neural Networks, particularly in the fields of semantic segmentation, object detection, and visual question answering, greatly augments the disaster management capabilities of drones. Through Deep Learning algorithms, drones can swiftly pinpoint and categorize specific objects or structures, such as compromised buildings or critical infrastructure, aiding in the prioritization of response efforts and optimal resource allocation.

This thesis delves into the fundamental principles and methodologies underpinning image semantic segmentation and object detection and Visual Question Answering, pivotal for bolstering the efficacy and precision of disaster management endeavors. Notably, models such as UNet, DeepLab, YOLOv8 and GPT2 showcase exceptional proficiency in accurately identifying and classifying objects within drone-captured imagery. The following synopsis outlines the principal themes explored within this master's thesis:

In the first chapter, we explore the definition of a disaster and provide examples of various types, such as earthquakes, floods, and forest fires. We delve into the concept of Disaster Management, defining its purpose and significance. Additionally, we discuss the definition of drones and why they are utilized in Disaster Management. We present examples of Disaster Management Drone applications and discuss the CNNs importance for Floods management. Lastly, we delve into the analysis of drone-captured images for surveillance purposes, including techniques such as semantic segmentation, image classification, object detection, and Visual Question Answering.

In the second chapter, our focus will be on two significant tasks within computer vision: semantic segmentation and object detection then we will pay attention to Natural Language Processing more precisely Visual Question Answering. We will delve into these tasks and explore a range of techniques and approaches employed to tackle these challenges using deep learning methods. Throughout the chapter, we

will discuss and analyze popular architectures such as UNet, DeepLabV3+, YOLO v8, and fine-tuning GPT2 and BERT. By examining these architectures, we will uncover their respective strengths and weaknesses.

In the Third chapter, we focus on the results of experiments of semantic segmentation, object detection, and Visual Question Answering techniques to drone images. We begin by defining the problem we aim to solve in this context. Next, we provide a detailed description of the dataset we will use. then delve into the network design details employed to solve the problem and how they were trained. Following that, we conclude by describing our experiments, validations, and results.

In the Final chapter, we present the web application that facilitates semantic segmentation, object detection, and Visual Question-answering models Training, evaluation, and prediction. the application starts with an index interface that represents the services that we provide, then we focus on describing each view by its functionalities. Finally, we describe some of the most popular deep learning frameworks and tools that we used in this application.

Chapter 1

Computer vision in Disasters Management

1.1 Introduction

Recently, the frequency and intensity of natural disasters have surged, highlighting the necessity for prompt and precise damage assessments to minimize the impact on lives and economies. Notably, the United States recorded significant economic losses from multiple natural disasters within a few years, emphasizing the urgent need for efficient response mechanisms. Traditional methods of damage assessment, often hampered by the inaccessibility of affected zones, are increasingly complemented by advanced technologies such as UAVs and satellite imagery. These tools provide high-resolution visuals, offering a clearer picture of the damage extent, which is crucial for strategic planning and resource allocation [1].

The integration of deep learning with aerial imagery has opened new avenues for analyzing disaster impacts, allowing for the automatic detection and classification of damages. This blend of UAV technology and artificial intelligence has the potential to drastically improve disaster management, making damage assessments faster and more accurate than ever before. As the world grapples with the escalating challenge of natural disasters, leveraging these advanced technologies is key to enhancing disaster response and recovery efforts, ultimately aiming to reduce both human and financial losses [1].

This chapter will make terms, concepts, and approaches clearer using solid definitions, details, and examples.

1.2 Disasters Overview

1.2.1 Types of Disasters

Disasters can be broadly categorized into two main types: natural disasters and man-made disasters. Each type encompasses various events that can cause significant harm to environments, economies, and populations.

1.2.1.1 Human-Caused Disasters

Man-made disasters are catastrophic events arising from human activities that adversely affect both the environment and society. These activities typically involve technological failures, industrial accidents, and inadequate environmental stewardship, reflecting a disregard for the natural world. Unlike natural disasters, which are prompted by natural phenomena, man-made disasters stem directly from human actions such as errors, negligence, or the failure to prudently manage interactions with the environment [2].

Examples of man-made disasters include industrial accidents, pollution, deforestation, and urban sprawl, each contributing to environmental decline and heightening the risk of further disasters. These instances underscore the critical need for responsible environmental management and preventive measures to safeguard against such catastrophic outcomes [2].

1.2.1.2 Natural Disasters

Natural disasters are categorized into various types based on their origin and inherent characteristics. These extreme events, stemming from Earth's natural processes, significantly affect human societies and natural ecosystems. The process of risk analysis for these disasters involves evaluating their likelihood of occurrence, potential impacts, and the effectiveness of response and mitigation strategies. Such a detailed classification is crucial for comprehensively understanding the multifaceted nature of natural disasters and their disparate effects across diverse regions and communities [3].

1.2.2 Examples of Natural Disasters

1.2.2.1 Earthquakes

Earthquakes cause immediate destruction to infrastructure, severely disrupting normal living conditions and healthcare services. For example, the earthquake in L'Aquila, Italy, not only damaged buildings but also led to significant public health issues, including a Salmonella outbreak among 155 children, attributed to contaminated water supplies. The sudden nature of earthquakes complicates timely emergency response and increases the vulnerability of affected populations to infectious diseases [4]. The figure 1.1 shows an example of an earthquake in the Turkish city of Hatay.

Figure 1.1: An aerial view of collapsed buildings in the Turkish city of Hatay [5]

1.2.2.2 Forest fires

Forest fires are devastating events that cause the loss of lives and serious damage to both nature and human properties, including thousands of hectares of forest and hundreds of homes. These fires are very harmful to healthy forests and the environment. They cause severe and often permanent damage, adding 30% of the carbon dioxide (CO_2) in the atmosphere, along with large amounts of smoke and other pollutants. Over time, they negatively affect local weather, contribute to global warming, and lead to the extinction of rare plants and animals. The figure 1.2 shows an example of a Forest fire in the Mid-Atlantic.

Figure 1.2: Forest fire in a Mid-Atlantic.
[6]

1.2.2.3 Floods

Floods are among the most common natural disasters in Europe and carry a high risk of spreading infectious diseases. They typically result in widespread water contamination and disruption of sanitary conditions, leading to diseases such as leptospirosis. Several instances highlight how floods have increased the incidence of infectious diseases. For example, in Bulgaria, heavy rainfall in 2014 led to a spike in leptospirosis cases, from 12 in 2010 to 30 in 2014. The persistent and expansive nature of floodwaters makes managing these disasters particularly challenging [4]. The figure 1.3 shows an example of a neighborhood in Jackson, Kentucky, overwhelmed by flash flooding.

Figure 1.3: A neighborhood in Jackson, Kentucky, is overwhelmed by flash flooding after heavy rains caused the Kentucky River to overflow in July 2022
[7]

1.2.3 Disaster Management

1.2.3.1 The Disaster Management Cycle (DMC)

The Disaster Management Cycle (DMC) shown in the figure 1.4 is a comprehensive framework used to manage and mitigate the impacts of disasters. It consists of four interlinked stages that facilitate a systematic approach to disaster risk management. Each stage plays a crucial role in ensuring the resilience and recovery of affected communities [8].

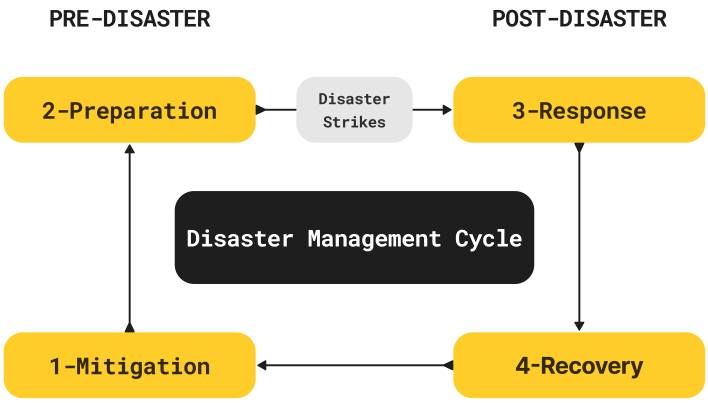


Figure 1.4: Disaster Management Cycle (DMC).

Here is an elaboration of each stage :

- **Mitigation**: The initial phase of the DMC involves implementing proactive measures to prevent the occurrence of disasters or to reduce the community's vulnerability. Strategies employed during this phase include constructing flood defenses, enforcing strict land-use planning laws, and adhering to elevated building standards. The primary objective is to minimize both human and material losses before any disaster unfolds [8].

- **Preparation**: This stage equips the community to respond effectively in the event of a disaster. It encompasses the training of emergency services, public education on disaster response, and the development of comprehensive evacuation and emergency communication plans. These preparations are vital for ensuring a rapid and efficient response that can mitigate the disaster's effects [8].

- **Response**: Activated during a disaster, this phase focuses on the immediate mobilization of emergency services and first responders to safeguard lives and provide essential services. Key actions include delivering medical care and supplying food, water, and shelter to affected individuals. The response phase is critical and demands swift, coordinated actions to reduce the overall impact of the disaster [8].

- **Recovery**: The final stage aims to restore the affected area to its pre-disaster conditions. This involves the reconstruction of infrastructure, restoration of essential services, and assistance in the return of displaced populations. Recovery efforts may also extend to include the enhancement of local policies and practices to improve future resilience against similar events [8].

1.3 Aerial Imagery

Aerial imagery includes photographs and visual data captured from airborne platforms, such as airplanes, helicopters, or UAVs. This form of imagery is a crucial tool for providing expansive views from above, essential for a variety of applications from environmental monitoring to urban planning and strategic surveillance.

1.3.1 Applications and Utility

Aerial imagery is fundamental to geographic mapping, delivering detailed visual representations that assist in land use planning and development control. It also plays a critical role in environmental monitoring, enabling researchers to observe temporal changes in ecosystems, such as vegetation dynamics, deforestation rates, and urban expansion. Furthermore, in the realm of surveillance, both governmental and non-governmental entities utilize aerial imagery for border security, emergency response, and monitoring of sensitive locations [9], [10].

1.3.2 Satellite Remote Sensing

Satellites, sophisticatedly engineered to orbit the Earth, are equipped with sensors to collect data about the Earth's surface. They provide essential insights for global and environmental monitoring through continuous time-series data. Satellites function at various resolutions and temporal frequencies, broadly categorized into high-orbit and low-orbit types. High-orbit satellites, such as MODIS or geostationary satellites, offer high temporal frequency data but at coarser spatial resolutions. In contrast, low-orbit satellites like Pleiades and Ikonos deliver very high spatial resolution data, though with more limited temporal coverage [9], [10].

1.3.3 Unmanned Aerial Vehicles (UAVs)

Unmanned aerial vehicles (UAVs), often colloquially known as drones, are compact aircraft designed to fly either autonomously or by remote control. Initially developed for military applications, they have since become a significant area of study. In addition to UAVs, these aircraft are also referred to as UAS (Unmanned Aerial Systems), Remotely Piloted Vehicles (RPVs), and Remotely Piloted Aircraft Systems (RPAS). UAVs are useful in a variety of applications and can be used for crime scene surveillance, habitat destruction assessment [1], and even in disaster areas. In the figure 1.5, we present an overview of the proposed DisasterVision system for disaster management using UAVs technology.

.

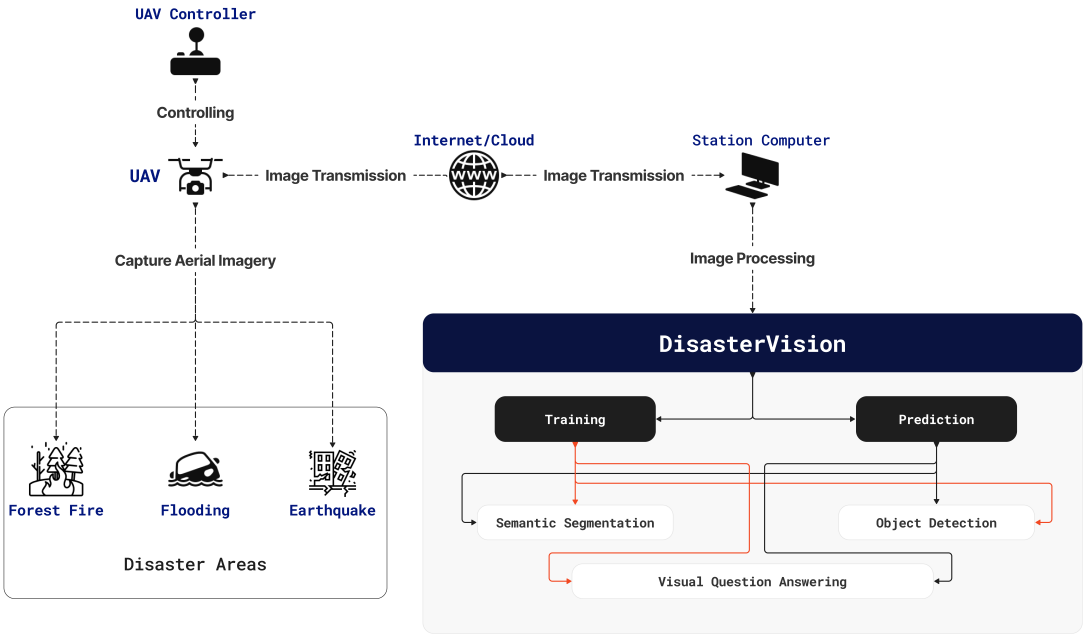


Figure 1.5: Unmanned Aerial Vehicles for Disaster Management

1.4 UAVs, Computer Vision and Deep Learning for Flood Management

1.4.1 Case Studies in UAV and Computer Vision Applications

Several case studies highlight the successful application of UAVs and computer vision in disaster management. For instance, after floods or earthquakes, UAVs have been deployed to capture detailed imagery of the impacted areas, which is then analyzed using computer vision techniques to map damage and identify zones requiring urgent attention as shown in the figure 1.6. These practical applications underscore the potential of integrating UAVs and computer vision to enhance the effectiveness of disaster response and recovery efforts [1].

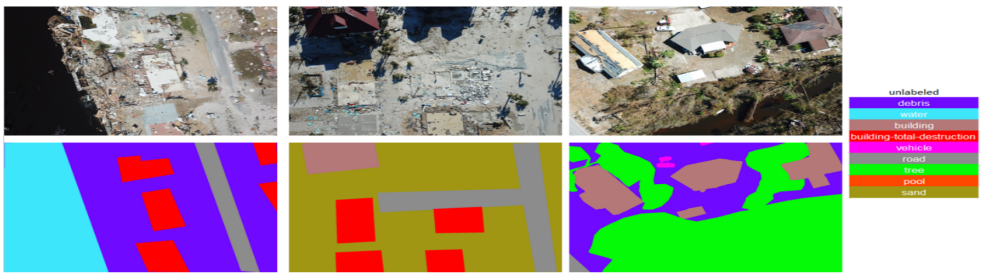


Figure 1.6: : Illustration of complex scenes of HRUD dataset. The first row shows the original image and the second row shows the corresponding annotations [11]

1.4.2 Flood Management Through UAVs

UAVs are employed extensively for capturing high-resolution imagery of flood-affected regions. Their ability to collect real-time data and operational flexibility make UAVs indispensable in rapid disaster response scenarios. UAVs facilitate the efficient monitoring of extensive areas impacted by floods, delivering critical data essential for an accurate and timely assessment. The deployment of UAVs in this context significantly accelerates the assessment process and enhances the safety of response personnel by minimizing their exposure to hazardous conditions [1].

1.4.3 Application and Effectiveness of Deep Learning

The application of deep learning in flood management involves processing UAV-captured imagery to perform accurate semantic segmentation of the affected areas. This segmentation is crucial as it aids in determining the extent of the flooding, categorizing the impacted zones, and consequently supporting the management and mitigation strategies following the disaster. The integration of UAVs with the DeepLabV3 algorithm enables detailed terrain classification and the identification of critical areas that require immediate intervention. This synergy between aerial data collection and advanced image processing techniques enhances the effectiveness of disaster management operations, providing a robust framework for rapid response and strategic planning in the aftermath of floods [1].

1.5 Objectives and Contributions of the Study

This study is dedicated to the development of DisasterVision, an innovative computer vision system aimed at revolutionizing disaster management with a particular emphasis on flood disasters. By harnessing the capabilities of advanced computer vision techniques and utilizing the rich data provided by the FloodNet dataset [1], DisasterVision aspires to significantly enhance the efficiency and accuracy of analyzing flood scenes. This section details the study's objectives, anticipated contributions, and potential impact on disaster response and recovery efforts.

1.5.1 Perspectives of DisasterVision

The primary objective of this study is the creation of DisasterVision, a system that embodies the integration of cutting-edge computer vision technologies to offer a holistic approach to disaster management. DisasterVision will be designed to:

- Provide rapid and precise semantic segmentation of flooded areas, facilitating immediate and informed decision-making, as demonstrated by the work on flooded area segmentation using UAV images [1].
- Implement object detection algorithms which involve identifying the specific class to which an object belongs and predicting its location by defining a bounding box around it to identify and assess damage to infrastructure and natural landscapes [1].

- Utilize Natural Language Processing (NLP) for Visual Question Answering (VQA) to enable intuitive interaction with the disaster data, allowing for more efficient disaster scene analysis.

1.5.2 Expected Contributions to Disaster Management

The development of DisasterVision is expected to make significant contributions to the field of disaster management, including:

- Advancing the state-of-the-art in computer vision applications for disaster response, specifically in the context of flood disaster management, by leveraging the comprehensive capabilities of UAV imagery for natural disaster damage assessment [1].
- Providing emergency response teams with a powerful tool for rapid and accurate flood impact assessment, facilitating quicker and more targeted response actions.
- Contributing to the body of computer vision and disaster management knowledge through empirical research and development.

1.6 Conclusion

In this chapter, we have explored various disasters and the importance of quick and precise damage management. Traditional methods often fall short, prompting the adoption of advanced technologies like UAVs for capturing high-resolution images. We also discussed both man-made and natural disasters, such as earthquakes, and floods, and introduced the Disaster Management Cycle (DMC) for effective response. Additionally, we explored how UAVs and deep learning can aid in flood management. Finally, we introduced DisasterVision and its role in accurate damage assessment, highlighting the significant impact of technology on enhancing disaster response and recovery.

Chapter 2

Deep Learning Approaches

2.1 Introduction

Deep learning has revolutionized artificial intelligence, particularly in computer vision tasks. Techniques like semantic segmentation, object detection, and visual question answering (VQA) play pivotal roles. Semantic segmentation divides images into meaningful segments, object detection identifies and classifies objects, while VQA enables machines to answer questions about images. In this chapter, we explore the applications and methodologies behind these deep learning techniques.

2.2 Semantic Segmentation

Semantic segmentation is a crucial task in computer vision, aiming to partition an image into meaningful regions corresponding to different object classes. In recent years, deep learning-based approaches have significantly advanced the field of semantic segmentation, providing highly accurate and efficient solutions for a wide range of applications, from medical imaging to autonomous driving.

2.2.1 U-Net

The U-Net architecture has gained significant popularity in image segmentation tasks, particularly within the biomedical field. It was first introduced in a 2015 paper authored by Olaf Ronneberger, Philipp Fischer, and Thomas Brox [12].

The U-Net architecture illustrated in Figure 2.1 is specifically tailored for image segmentation tasks, aiming to categorize each pixel in an image into predefined categories. It comprises two distinct paths: the first is known as the contracting path, encoder, or analysis path, while the second is referred to as the expansion path, decoder, or synthesis path [13].

The encoder path of the U-Net architecture comprises a sequence of convolutional and max-pooling layers. These layers progressively diminish the spatial dimensions of the input image while augmenting the number of channels. This sequential process enables the network to capture higher-level features of the image, such as edges, corners, and textures.

On the other hand, the decoder path employs a series of up-convolutional and concatenation layers to enhance the spatial resolution of the feature maps while reducing the channel count. The incorporation of skip connections between the encoder and decoder paths facilitates the retrieval of detailed information from earlier layers, thereby enhancing segmentation accuracy.

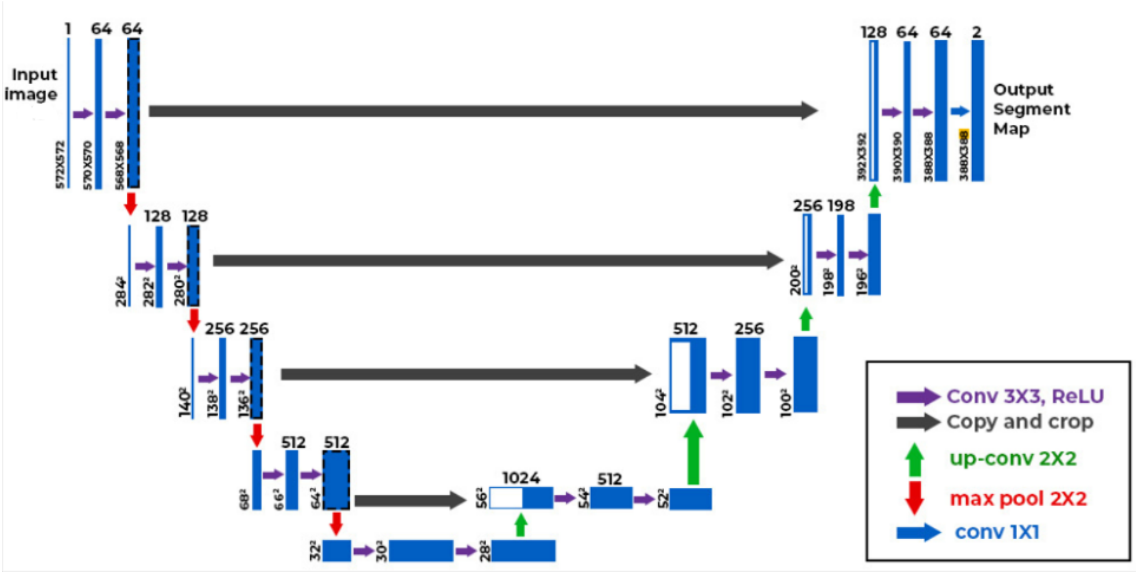


Figure 2.1: the U-Net architecture for image segmentation [11]

- **Input Image Tile:** The U-Net model takes a small portion of an image, known as an input image tile, as its input.
- **Convolution with 3x3 Filter and ReLU Activation:** This operation applies a 3x3 convolutional filter to the input image tile, followed by a rectified linear unit (ReLU) activation function, aiding in feature extraction.
- **Copy and Crop:** After each convolutional layer in the contracting path, the feature maps are duplicated and merged with corresponding feature maps in the expanding path. Before merging, the expanding path's feature maps are trimmed to match the size of those in the contracting path.
- **Max Pooling with 2x2 Filter:** This process decreases the spatial resolution of feature maps while retaining critical features. It's employed in the contracting path for downsampling.
- **Up-Convolution with 2x2 Filter:** In the expanding path, this step increases the spatial resolution of feature maps while reducing their channel count, aiding in upsampling.
- **Convolution with 1x1 Filter:** This operation blends channels in feature maps, diminishing dimensionality while enhancing representational capability.
- **Output Segmentation Map:** The U-Net model's final output is a segmentation map, assigning labels to each pixel in the input image tile based on class predictions.

2.2.2 DeeplabV3+

Deeplabv3+ was introduced in 2018 [14]. Its architecture adopts an encoder-decoder structure (see Figure 2.2), integrating atrous separable convolution. This convolutional approach combines depthwise convolution, performing a spatial convolution for each input channel, with pointwise convolution, which employs a 1x1 convolution using the output of the depthwise convolution.

DeepLabv3+ stands out as one of the premier semantic segmentation algorithms today. Building upon the foundation of DeepLabv3, this algorithm enhances its capabilities by introducing a streamlined and efficient decoder [15]. The network architecture of DeepLabv3+ is depicted in Figure below

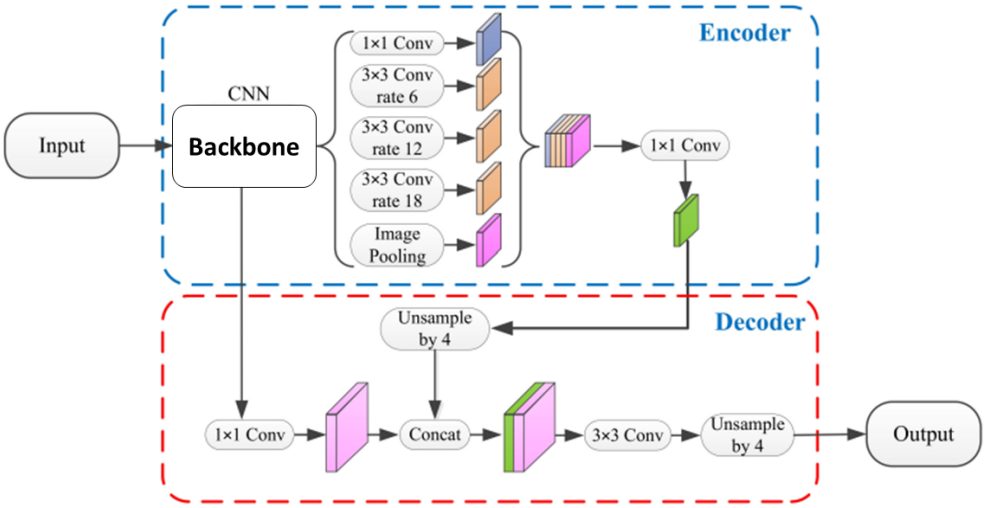


Figure 2.2: Deeplabv3+ architecture for image segmentation [12]

As depicted in Figure 2.2, the encoding phase of DeepLabv3+ primarily consists of two components: the backbone network and the atrous spatial pyramid pooling (ASPP) module. The decoding phase involves upsampling the output feature map from the encoding phase and merging it with a low-dimensional feature map. Subsequently, bilinear interpolation is applied to restore the input image size, yielding the final segmentation result [15].

The atrous spatial pyramid pooling module adopts a parallel structure, employing atrous convolution with varying atrous rates to capture multi-scale information effectively. Within the DeepLabv3+ network structure, the ASPP module encompasses a 1x1 convolution, three 3x3 atrous convolutions with atrous rates of 6, 12, 18, and a global average pooling branch. Each operation is followed by a batch normalization layer for data normalization. The feature maps produced by each branch are concatenated, and finally, a 1x1 convolution is applied to compress and integrate features [15].

The decoding part's structure is relatively straightforward. The final output of the encoding section of the network is a high-dimensional feature map with 256

channels. This feature map is then sent to the decoder. Initially, it's up-sampled using bilinear interpolation and then concatenated with low-level features of the same resolution from the backbone network.

To avoid the influence of the larger number of channels in the low-level features on the semantic information in the high-level features, a 1 x 1 convolution is applied to the low-level features to reduce their channels before concatenation. The concatenated features then pass through a 3 x 3 convolution to refine them. Finally, they undergo bilinear interpolation four times for up-sampling to restore them to the original image size, producing the final detection result.

The encoder-decoder structure implemented in DeepLabv3+ is a notable innovation. The encoder is mainly used to encode rich context information, while the concise and efficient decoder is used to restore the boundary of the detected object. Moreover, the network employs atrous convolution to enable feature extraction by the encoder at various resolutions, enhancing the balance between detection speed and accuracy.

2.2.3 Performance Metrics

After developing a deep learning model for semantic segmentation, the next step is to assess its predictive performance. This involves partitioning the data into training, validation, and test sets to calculate relevant metrics. These metrics include:

Accuracy

This metric quantifies the percentage of correct predictions made by the model. It's calculated by dividing the number of correct predictions by the total number of predictions. The formal definition of accuracy is:

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

IoU

The Intersection-Over-Union (IoU) method is frequently employed to evaluate the performance of image segmentation techniques. This method computes the IoU for each semantic class and then averages over all classes, making it a standard assessment metric for semantic image segmentation [16].

Mean IoU is a variant of IoU used specifically for semantic segmentation. It calculates the IoU for each semantic class separately before averaging over all classes. It's defined as the ratio of true positive predictions to the sum of true positive, false positive, and false negative predictions for each class. IoU is defined as follows:

$$\text{IoU} = \frac{TP}{TP + FN + FP}$$

where, TP, FP, and FN denote the true positive, false positive and false negative counts, respectively.

2.3 Objects Detection

Object detection is a key task in computer vision, involving identifying and localizing objects within images or videos by outputting bounding boxes around detected objects, as shown in Figure 2.3 . It enables computers to understand visual information like humans, recognizing objects despite variations. Historically reliant on manual feature design and simple models, object detection has advanced with deep learning, particularly Convolutional Neural Networks (CNNs), which automate feature extraction and enhance accuracy. Modern methods such as R-CNN, Fast R-CNN, Faster R-CNN, YOLO, and SSD have set new performance benchmarks. Innovations like the Focal Loss function address challenges like class imbalance and real-time processing needs. Object detection is crucial for applications in autonomous driving, traffic monitoring, security surveillance, and human-computer interaction. The field is expected to grow, incorporating sophisticated neural architectures and learning schemes for complex real-world scenarios [17] , [18].

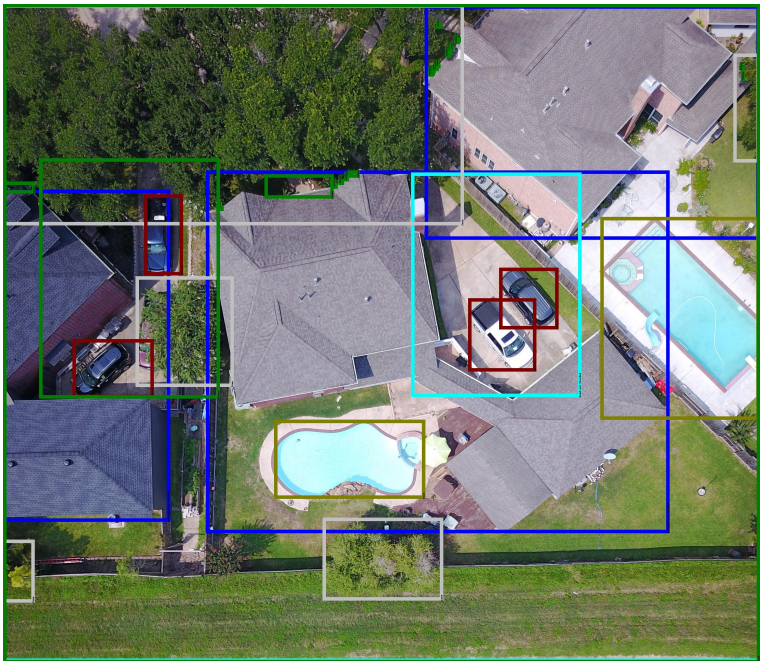


Figure 2.3: Example of predicted bounding boxes consolidated into an inference image.

2.3.1 Object Detection Approaches in Deep Learning

Object detection is a pivotal area in computer vision, significantly enhanced by deep learning technologies. These techniques have evolved from traditional machine learning approaches to sophisticated deep learning models that improve accuracy and efficiency in detecting objects across various domains. Below, as shown in Figure 2.4 , we explore some key techniques and approaches for object detection using deep learning.

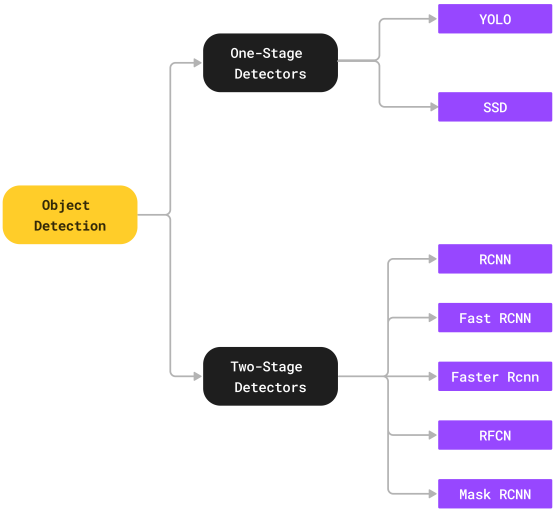


Figure 2.4: Object detection stages

2.3.1.1 Two-Stage Detectors

Two-stage detectors, such as R-CNN and its variants (Fast R-CNN, Faster R-CNN), have set a high standard in object detection accuracy. These methods work in two phases: first, they generate regions or proposals where objects might be located, and second, they classify each proposal into different object classes [18].

2.3.1.2 One-Stage Detectors

One-stage detectors such as YOLO (You Only Look Once) and SSD (Single Shot MultiDetector) provide a faster alternative by eliminating the proposal generation phase and detecting objects in a single pass through the network. Despite their speed, these methods have traditionally struggled with accuracy compared to two-stage methods. However, with improvements such as the use of feature pyramids and focal loss, one-stage detectors are closing the accuracy gap while offering significant speed advantages [18].

2.3.2 YOLO (You Only Look Once)

YOLO fundamentally changes the approach to object detection by framing it as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Unlike traditional methods that apply a classifier to different regions of an image, YOLO simultaneously predicts multiple bounding boxes and class probabilities for those boxes. This approach allows YOLO to achieve high speeds and accuracy, making it suitable for applications requiring real-time processing [19].

2.3.3 YOLO Process for Object Detection

2.3.3.1 Image and Grid Setup

- **Input and Preprocessing**: YOLO processes images in a single forward pass using a deep convolutional neural network. It first resizes the input image to a fixed dimension (e.g., 416x416 pixels for certain versions), ensuring uniformity for efficient processing [20].
- **Grid Division**: The image is divided into an SS grid (where S is the number of cells in the grid) as shown in Figure 2.5 . Each cell in the grid is responsible for predicting objects whose center point falls within the cell. This division simplifies the problem by spatially segregating the predictions [20].

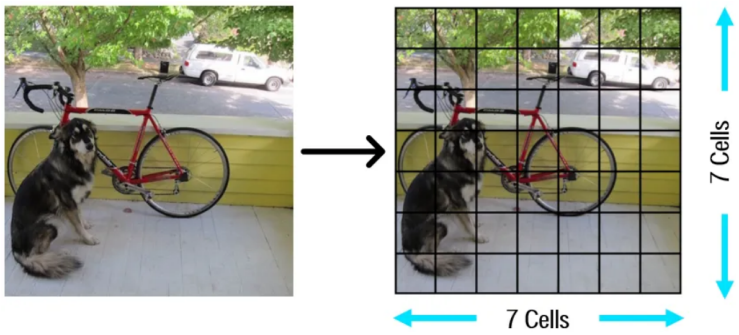


Figure 2.5: Divide the input image to an SS grid
[21]

2.3.3.2 Prediction of Bounding Boxes and Probabilities

- **Bounding Box Predictions**: Each grid cell predicts multiple bounding boxes. For each box, the model predicts four coordinates (center x, center y, width, height), normalized relative to the cell's location, and a confidence score that indicates the presence of an object and how well the predicted box fits the object [20].
- **Class Probabilities**: Each cell predicts the probabilities of various classes for the objects it detects. These probabilities are conditioned on the cell containing an object. The combination of bounding box predictions and class probabilities allows YOLO to localize and identify objects in a single pass through the network [20].

  Figure 2.6 provides a visual representation of the process involved in predicting bounding boxes and class probabilities in object detection.

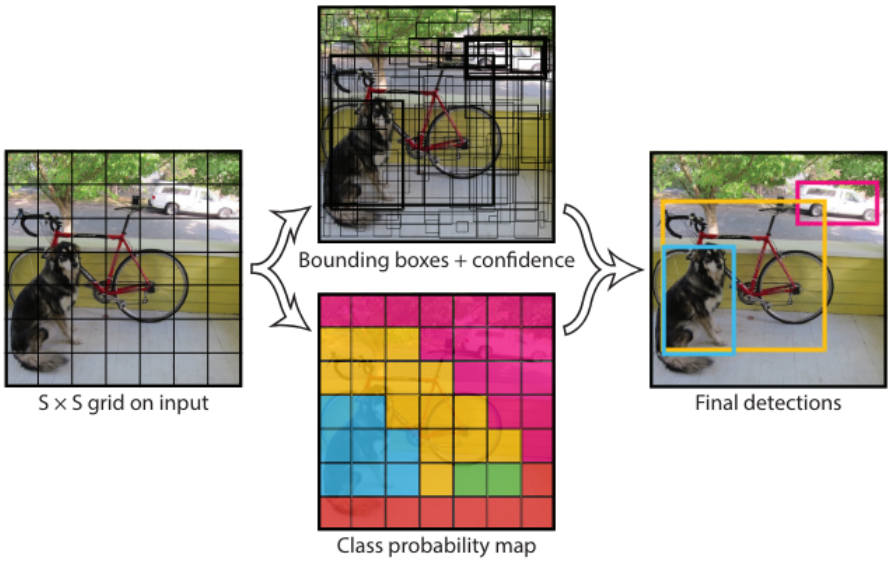


Figure 2.6: Prediction of Bounding Boxes and Probabilities
[22]

2.3.3.3 Model Architecture and Output

- **Network Design**: YOLO uses a single convolutional network that simultaneously predicts multiple bounding boxes and class probabilities for those boxes. The design includes features like anchor boxes, which help the model learn to predict correct box sizes based on the dataset it was trained on [20].
- **Output**: The output from YOLO is typically a tensor of shape $[SS(B5+C)]$ as shown in Figure 2.7 , where B is the number of bounding boxes each cell predicts, 5 represents the value of the four bounding box coordinates plus the confidence score, and C is the number of class probabilities [20].

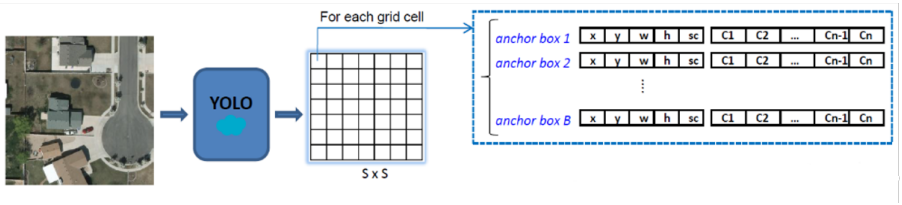


Figure 2.7: The output tensor shape of YOLO
[23]

2.3.3.4 Post-processing via Non-Max Suppression

- **Refining Predictions**: To reduce the number of overlapping boxes and improve the model's precision, YOLO employs non-max suppression. This process filters out boxes that have a high overlap (as defined by IoU) with a stronger predicted box [20].

2.3.3.5 Loss Function During Training

- **Box Loss**: This loss measures the accuracy of the predicted bounding boxes' locations and dimensions relative to the ground truth [20] . It is calculated using the sum of squared errors for the coordinates and dimensions:

$$\text{Box Loss} = \sum_{i \in \text{pos}} \lambda_{\text{coord}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

  Here, x, y, w, h denote the ground truth coordinates and dimensions, $\hat{x}, \hat{y}, \hat{w}, \hat{h}$ are the predicted values, and λ_{coord} is a scaling factor that adjusts the importance of this loss component.

- **Classification Loss**: This component penalizes incorrect class predictions using cross-entropy loss between the predicted class probabilities and the ground truth labels:

$$\text{Classification Loss} = -\sum_{i \in \text{pos}} \sum_{c=1}^{C} y_{i,c} \log(\hat{y}_{i,c})$$

  In this formula, C represents the number of classes, $y_{i,c}$ is the ground truth label for class c (1 for the actual class, 0 otherwise), and $\hat{y}_{i,c}$ is the predicted probability of class c for the i-th bounding box [20].

- **Confidence Loss**: This loss calculates the error in the predicted confidence scores for the bounding boxes, penalizing both overconfident false predictions and underconfident true predictions:

$$\text{Confidence Loss} = \sum_{i \in \text{pos}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{j \in \text{neg}} (C_j - \hat{C}_j)^2$$

  C_i and $\hat{C}_i$ are the ground truth and predicted confidence scores for bounding boxes that overlap with ground truth objects (positives), while C_j and $\hat{C}_j$ are for those that do not (negatives), with λ_{noobj} scaling down the influence of negative examples [20].

2.3.3.6 Accuracy Metrics During Training

- **Precision and Recall**: These are fundamental metrics derived from the confusion matrix, reflecting the proportion of correct positive predictions and the model's ability to capture all relevant instances, respectively:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

  Here, TP, FP, and FN stand for true positives, false positives, and false negatives, respectively [20].

- **Intersection over Union (IoU)**: This metric quantifies the overlap between the predicted and ground truth bounding boxes:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

- **Mean Average Precision (mAP)**: This is a comprehensive performance metric, calculated by averaging the area under the precision-recall curve for each class and each IoU threshold:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AUC}_{\text{PR}}(c)$$

C is the number of classes and $\text{AUC}_{\text{PR}}(c)$ is the area under the curve for class c [20].

2.4 Natural Language Processing

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language. It involves the development of algorithms and models to understand, interpret, and generate human language data. NLP has numerous applications, including machine translation, sentiment analysis, text summarization, and question answering. Visual Question Answering (VQA) is a way for computers to answer questions about what they see in pictures or videos using words. It mixes together computer vision, which helps machines understand images, with natural language processing, which helps them understand and generate text-based responses [24].

2.4.1 Pre-Processing Techniques

The Text pre-processing step is simply means bringing the document to a format that is easily understandable, predictable, and analysable by the machine through the various machine learning algorithms [25] while The objective of Visual Question Answering (VQA) is to extract semantics relevant to the question from images. This encompasses tasks ranging from detecting intricate details to inferring abstract scene attributes across the entire image, all based on the specific question posed [26]. Some of the widely used pre-processing techniques are:

2.4.1.1 Tokenization

Tokenization involves breaking sentences into individual units such as words, characters, and punctuation marks, each of which is referred to as a token. The splitting process typically occurs at spaces or punctuation marks. This initial step aids in removing unnecessary words during subsequent processing stages [25].

2.4.1.2 StopWords Removal

Common words like "the", "are", "is", and "and" typically lack significance in natural language processing, except in specific contexts. For instance, in text or document classification tasks, these words often don't carry much weight, and only keywords relevant to the topics are extracted. Therefore, identifying and removing these stopwords can enhance the performance of classification algorithms. However, it's essential to note that in certain scenarios, such as conversational models, negation words like "no", "cannot", "won't", and "not" are crucial for understanding the context and intent of sentences [25].

2.4.1.3 Lemmatization

Lemmatization involves either removing or replacing the suffix of a word to bring it to its base form, known as the lemma. Unlike stemming, the resulting lemma is always a meaningful word. Lemmatization is a widely used text preprocessing step in Natural Language Processing and has been shown to yield excellent results.

For instance, the lemma of the word "caring" is "care," which is a meaningful word [25].

2.4.1.4 Stemming

Stemming is a method of truncating words aggressively to their base root form by removing suffixes while attempting to preserve the semantic meaning across various forms of the word. However, this approach doesn't always yield accurate results because the word may lose its intended meaning.

2.4.2 Feature Extraction Techniques

Feature extraction of text involves converting features into vector forms to make them comprehensible to the machine. These techniques extract features and represent them as vectors, which are then inputted into classifier models. Some of the feature extraction techniques discussed here are :

2.4.2.1 Named Entity Recognition (NER)

Identifies entities such as names of people, organizations, and locations within the text. It functions similarly to POS tagging but specifically focuses on distinguishing and categorizing these named entities. NER is valuable for isolating nouns and providing a clearer understanding of names, places, and organizations mentioned. It serves as a foundational step in tasks where extracting named entities from documents is necessary, such as information retrieval [25].

2.4.2.2 The Bag of Words (BoW)

BoW model is a crucial technique for extracting features from text, which are then used by classifiers. It groups features together based on the frequency of word occurrences in the document, without considering their positions in the text [25].

The basic idea is that documents containing the same words are contextually similar. This model finds application in document classification, keyword matching, and similar tasks [25].

One issue with the BoW model is that it favors words with higher occurrence rates, potentially overemphasizing their importance[27]. However, some frequently occurring words may lack significant information for classification or clustering tasks. Additionally, longer documents naturally have higher occurrence rates, which can lead to decreased accuracy for the BoW model [25].

2.4.2.3 TF-IDF

TF-IDF is utilized to reduce the influence of frequent words that carry less significance in a document's context. It consists of two components: Term Frequency

(TF) and Inverse Document Frequency (IDF) [25].

TF measures the frequency of a word in a document while normalizing it by the document's length, ensuring fair accuracy for both small and large datasets[28]. This normalization technique divides the frequency by the document length [25].

On the other hand, IDF determines the importance of a word in the document by scaling down less important words. It achieves this by computing the logarithm of the ratio of total documents to the total occurrences of the word in that document [25].

$$\text{tfidf}_{i,j} = \text{tf}_{i,j} \log\left(\frac{N}{\text{df}_i}\right)$$

where :

- $\text{tf}_{i,j}$ = total number of occurrences of i in j
- df_i = total number of documents (speeches) containing i
- N = total number of documents (speeches)

2.4.3 VQA Modeling Techniques

For VQA models, it's crucial to comprehend both the image and the question. In natural language question understanding, Long Short-Term Memory (LSTM) networks and Gated Recurrent Unit (GRU) networks are commonly used. These models have a recurrent structure proven effective in modeling natural language sentences [29].

In image understanding, traditional approaches utilize VGG or ResNet to extract image features, as both achieve high performance in low-level computer vision tasks like image classification. However, VGG and ResNet split an image into regions that don't fully capture its natural semantics. Instead, they proposed using a Faster-RCNN+ResNet model to extract object-level region features, which provide a more natural representation of the image. This model has become one of the most popular approaches for image feature extraction in the VQA task [29].

2.4.3.1 Transformers Architecture

The majority of competitive neural sequence transduction models adopt an encoder-decoder structure. In this structure, the encoder maps an input sequence of symbol representations The tuple $(x_1, \ldots, x_n)$. to a sequence of continuous representations The vector $\mathbf{z} = (z_1, \ldots, z_n)$.Once z is obtained, the decoder generates an output sequence The tuple $(y_1, \ldots, y_m)$. of symbols one at a time. The model is auto-regressive, meaning it consumes previously generated symbols as additional input when generating the next one [30].

The Transformer follows this general architecture by employing stacked self-attention and point-wise, fully connected layers for both the encoder and decoder. These components are illustrated in the left and right halves of the Figure 2.8, respectively [30].

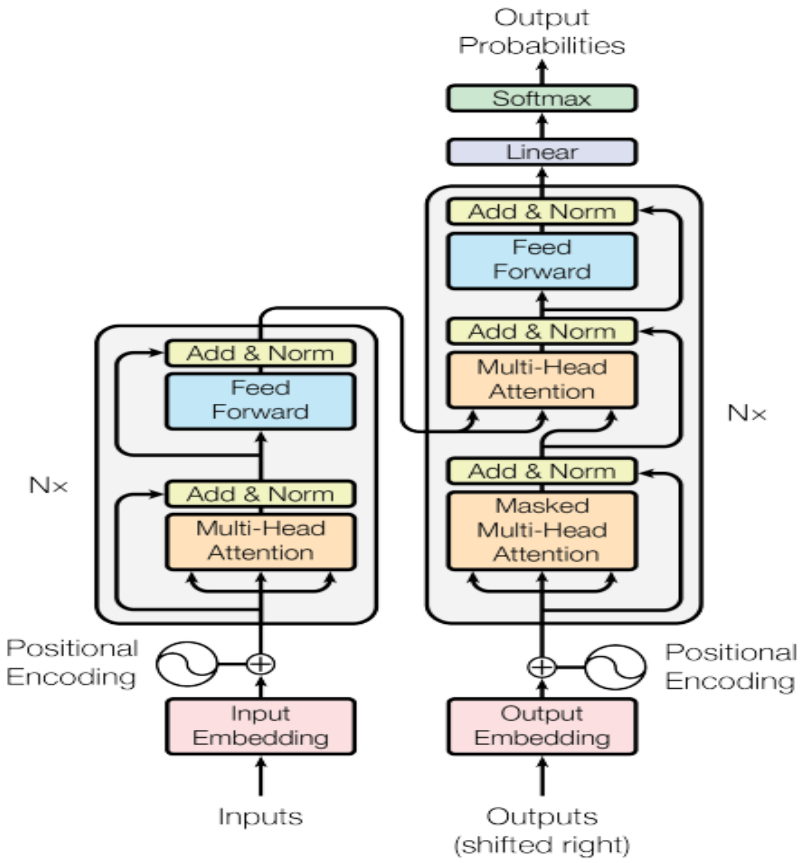


Figure 2.8: The Transformer - model architecture
[30]

Encoder:

The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position wise fully connected feed-forward network. We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is LayerNorm(x + Sublayer(x)), where Sublayer(x) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension d model = 512 [30].

Decoder:

The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with the fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i [30].

2.4.3.2 Attention

Attention can be described as a function that maps a query and a set of key-value pairs to an output, where all of these - query, keys, values, and output - are vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is determined by a compatibility function of the query with the corresponding key [29].

2.4.4 OpenAI GPT-2 model Fine Tuning

Recently, deep learning and pre-trained models have shown impressive performance across various language tasks. Specifically, fine-tuning pre-trained models like OpenAI's GPT (Generative Pre-Training), GPT-2, and BERT has become the standard approach for achieving state-of-the-art results. GPT-2, the successor to GPT, and BERT are both capable of text generation, but the experts found that GPT-2 generates higher-quality text. In fact, GPT-2 is considered so powerful that there's a high risk of its malicious use. To address this concern, OpenAI has decided to keep its largest GPT-2 model (1.5B parameters) closed, allowing more time for discussions about its potential impacts [31]. Figure 2.9 illustrates the structure of the applied GPT-2 medium architecture.

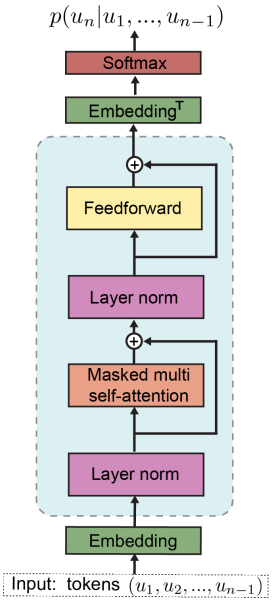


Figure 2.9: Structure of the applied GPT-2 medium architecture [32]

Transfer learning can be an effective strategy to adapt models to lower-resource languages by initially training a model for a source language and then further training (parts of) the model for a target language. GPT2 is an auto-regressive Transformer decoder-based language model for English and comes in four sizes: small (12 layers), medium (24 layers), large (36 layers), and extra large (48 layers) [31].

2.4.5 Google AI BERT Model

BERT, which stands for *Bidirectional Encoder Representations from Transformers*, is a groundbreaking language representation model introduced by Google AI. Unlike traditional language models that read text either left-to-right or right-to-left, BERT processes text bidirectionally, allowing it to understand the context of a word based on all surrounding words in a sentence. This bidirectional approach enables BERT to capture the nuanced meanings and relationships within the text, making it highly effective for a variety of natural language processing tasks such as question answering and language inference. In our study, we leverage BERT for preprocessing text, taking advantage of its ability to generate rich, context-aware embeddings that enhance the performance of downstream tasks [33].

2.5 Conclusion

In this chapter, we delved into the fundamental concepts and techniques for semantic segmentation, object detection and visual Question Answering in deep learning. We explored popular models like UNet, DeepLab, YOLO, GPT-2, and BERT, which have greatly influenced computer vision and NLP tasks. Given the rapid progress of deep learning, we anticipate the emergence of even more sophisticated models and techniques in the future, leading to more powerful and accurate deep learning applications than ever before.

Chapter 3

Experiments and Results

3.1 Introduction

In this chapter, we'll share our study on semantic segmentation, object detection, and visual question answering. First, we'll explain the problem we were trying to solve and describe the data we used. Then, we'll talk about the different methods we tried and how we trained our models. We'll go through the experiments we did and discuss the results, using specific measures to evaluate how well our models worked. At the end, we'll give some details about the hardware and software that helped us in this study.

3.2 Dataset

FloodNet offers high-resolution images captured from low-altitude sources, which sets it apart from satellite imagery taken from higher altitudes that may be obstructed by clouds and smoke. The distinct characteristics of FloodNet's images result in clearer scenes, which greatly aid deep-learning models in making more accurate decisions regarding post-disaster damage assessment. By providing detailed and unobstructed visual data, FloodNet improves the accuracy and effectiveness of deep learning algorithms in analyzing and assessing the extent of damage following a disaster. The data was collected using a small Unmanned Aerial System (UAS) platform, DJI Mavic Pro quadcopters, after Hurricane Harvey. The original dataset comprises 2393 images, split into 3 subsets 1445 for training,500 for validation, and 448 for testing.

The FloodNet dataset encompasses a range of classes or categories essential for semantic segmentation and object detection tasks. These classes are detailed in Table 3.1. By incorporating diverse classes, the dataset offers crucial insights into flooded regions, facilitating precise identification, segmentation, and detection of distinct objects or regions within the images. The color assignments associated with each class serve as visual cues, aiding in the differentiation of classes during both segmentation and object detection procedures.

These classes represent various objects or areas within the flooded scenes and offer valuable information for accurately identifying and assessing the extent of flooding and damages in the affected areas [34].

Class Index	Class Name
0	Background
1	Building flooded
2	Building non-flooded
3	Road flooded
4	Road non-flooded
5	Water
6	Tree
7	Vehicle
8	Pool
9	Grass

Table 3.1: FloodNet Dataset Classes

3.3 Semantic Segmentation

3.3.1 Dataset and Model Configuration

This section will present the dataset configuration utilized, and model hyperparametres, describe the selected backbone architecture, and explain the results obtained by applying the U-Net and DeepLabv3+ deep learning models to the FloodNet dataset.

For model training, approximately 60% of the available images from the dataset, which amounts to 1445 images with a resolution of 256x256 pixels, are used for training the models. The remaining 40% is reserved for the validation and test processes, which amounts to 948 images devised as 450 images for validation and 448 images for testing.

The training process involves using a batch size of 32 and running about 60 epochs. The number of steps per epoch is determined by dividing the number of training images by the batch size ($1445/32 = 45$).

additionally, the backbone is the architectural element that determines the arrangement of layers in the encoder network and establishes the structure of the decoder network. The backbone typically includes convolutional neural networks such as ResNet 34 and ResNet 50 which we have used in our models training.

The model output layer has been used as a softmax activation function, which is important for multi-category classification functions because it determines probability scores for each category.

AdamW Optimizer is used during the process of assembling models. As for the loss function we used the CrossEntropy function.

lastly, for the scheduler, we used the PolynomialLR with an initial Learning rate of 0,0001.

The model's performance is assessed using two metrics: accuracy, which measures the general correctness of predictions, and the Intersection over Union (IoU), which determines the overlap between predicted truth areas and the ground truth in the segmentation function.

To facilitate easy access without re-training the model from scratch, we store the models in .PTH format and save the training history, which includes metrics such as accuracy and loss, as a spy file once the training is complete. This allows us to

utilize the models for fine-tuning or to make a review of their performance at any desired time.

3.3.2 Models Evaluation and comparison

The performance metrics such as loss, accuracy, and mIoU are given in (Table3.2). In the training of our segmentation models, we have fixed the Hyperparameters of training so we can put all the models in the same training environment and evaluate them together.

Architecture + Backbone	Train_loss	Train_mIoU	Train_Acc	Val_loss	Val_mIoU	Val_Acc
Unet	0,185	0,589	0,935	0,378	0,377	0,888
DeepLabV3+	0.097	0.855	0.962	0.366	0.678	0.899

Table 3.2: Semantic Segmentation Results comparison

We can observe that the best results are achieved by the **Deeplabv3+** model. This approach gives an **mIoU** metric of **0.678**, which is the best among all the studied models. The same trend can be noted with the evaluation using the **accuracy** metric, where the highest value is achieved by the **Deeplabv3+** model is an accuracy with the value of **0.899**.

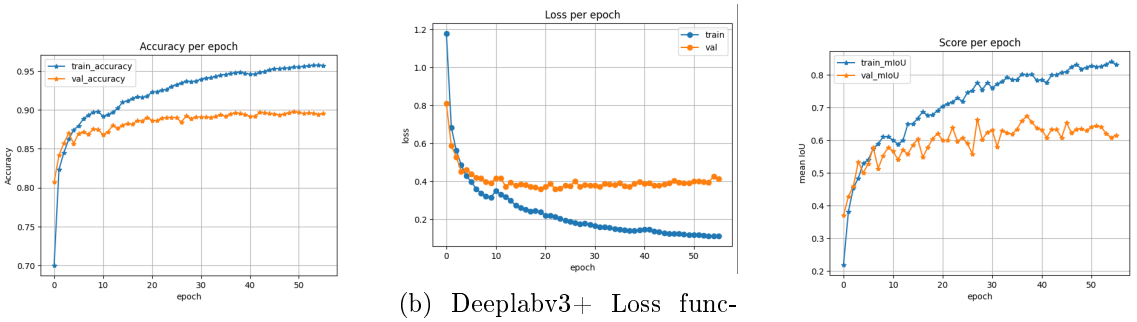
On the other side, we can see that the **Unet** model achieves slightly lower performance in terms of both **accuracy** and **mIoU** metrics, with an accuracy value of **0.888** and a **mIoU** value of **0.377**.

As a comparison result, we can say that the **Deeplabv3+** model outperforms the other model in terms of both metrics **accuracy** and **mIoU** metrics.

3.3.3 Performences plots

In the figures 3.2 , 3.1 we display the evolution of the values of the performance metrics in the first epochs of the training and validation process in our models (DeeplabV3+ and Unet) where the training was at the height of his tender.

DeepLabV3+



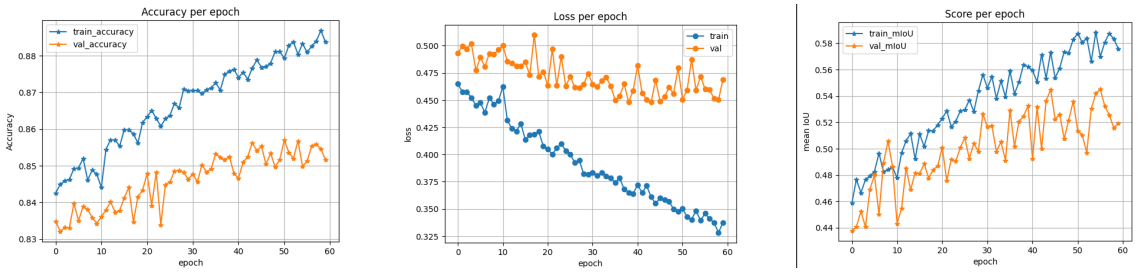
(a) Deeplabv3+ accuracy

(b) Deeplabv3+ Loss function

(c) Deeplabv3+ mean IoU

Figure 3.1: Values evolution of the performance metric using Deeplabv3+ model

Unet



(a) Unet accuracy (b) Unet Loss function (c) Unet mean IoU

Figure 3.2: Values evolution of the performance metric using Unet model

Deeplabv3+ Plots Analysing

- **Accuracy per Epoch**

  This plot shows the training accuracy (train-accuracy) and validation accuracy (val-accuracy) across epochs.

  Observations:

  Training Accuracy increases steadily, eventually plateauing around 0.95 after about 40 epochs.

  Validation Accuracy Rises quickly initially and then levels off around 0.88-0.89 after about 10 epochs.

  Analysis: The model demonstrates good generalization performance initially but starts overfitting after around 10 epochs. This is evidenced by the gap between training and validation accuracy, which widens as training progresses.

- **Loss per Epoch**

  This plot illustrates the training loss (train) and validation loss (val) over epochs.

  Observations:

  Training Loss decreases continuously, leveling off around 0.2 after about 40 epochs. Validation Loss drops sharply initially, then fluctuates and stabilizes around 0.4 after about 10 epochs.

  Analysis:

  The decrease in training loss indicates the model is learning effectively on the training data.

- **Score (Mean Intersection over Union - mIoU) per Epoch**

  This plot displays the mean Intersection over Union (mIoU) for training (train-mIoU) and validation (val-mIoU) data across epochs.

  Observations:

  Training mIoU: Increases steadily and reaches around 0.82 after about 50 epochs. Validation mIoU: Rises quickly and then oscillates around 0.65-0.68 from about epoch 10 onward.

Analysis:

The trend in mIoU is consistent with the accuracy and loss plots, showing good initial learning followed by overfitting.

Unet Plots Analysing

- **Accuracy per Epoch**

  Observations:

  Training Accuracy: Increases steadily, reaching around 0.88 by the end of the epochs.

  Validation Accuracy: Rises initially but fluctuates around 0.84 to 0.86 after approximately 10 epochs.

  Analysis: The steady increase in training accuracy indicates the model is learning well on the training data. The validation accuracy, however, shows a much flatter increase and significant fluctuation, which suggests that the model's performance on the validation data is not improving as consistently. The gap between training and validation accuracy widens over epochs, indicating the potential onset of overfitting.

- **Loss per Epoch**

  Observations:

  Training Loss: Decreases continuously and steadily throughout the training period.

  Validation Loss: Drops initially but then fluctuates and stabilizes around 0.45.

  Analysis:

  The continuous decrease in training loss indicates effective learning on the training data. The validation loss's fluctuation and stabilization around a higher value than the training loss further supports the presence of overfitting. The divergence between training and validation loss suggests that the model performs well on the training data but does not generalize as effectively to validation data.

- **Score (Mean Intersection over Union - mIoU) per Epoch**

  Observations:

  Training mIoU: Increases steadily, reaching around 0.58.

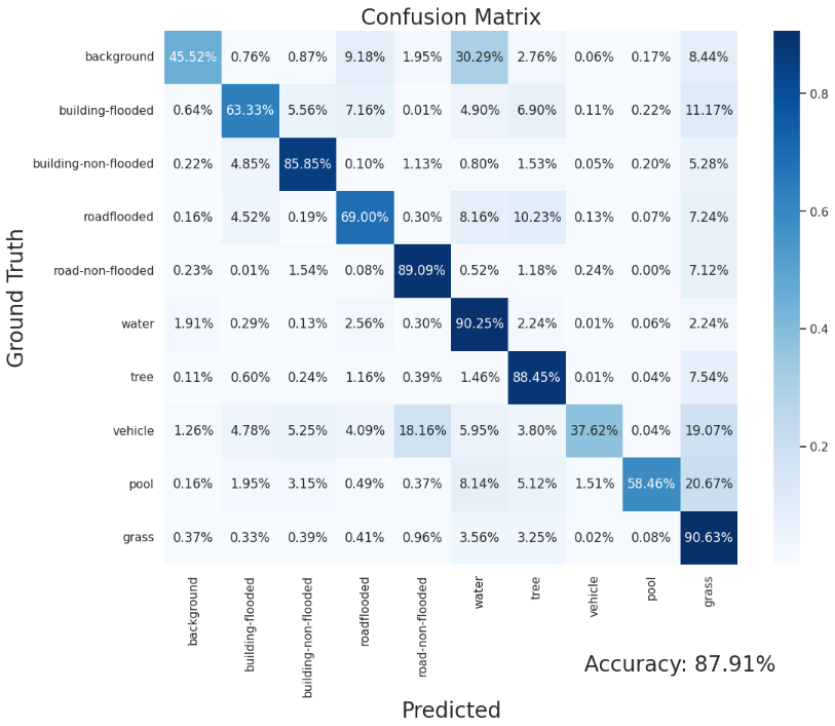
  Validation mIoU: Rises initially but then oscillates around 0.50 to 0.54.

  Analysis:

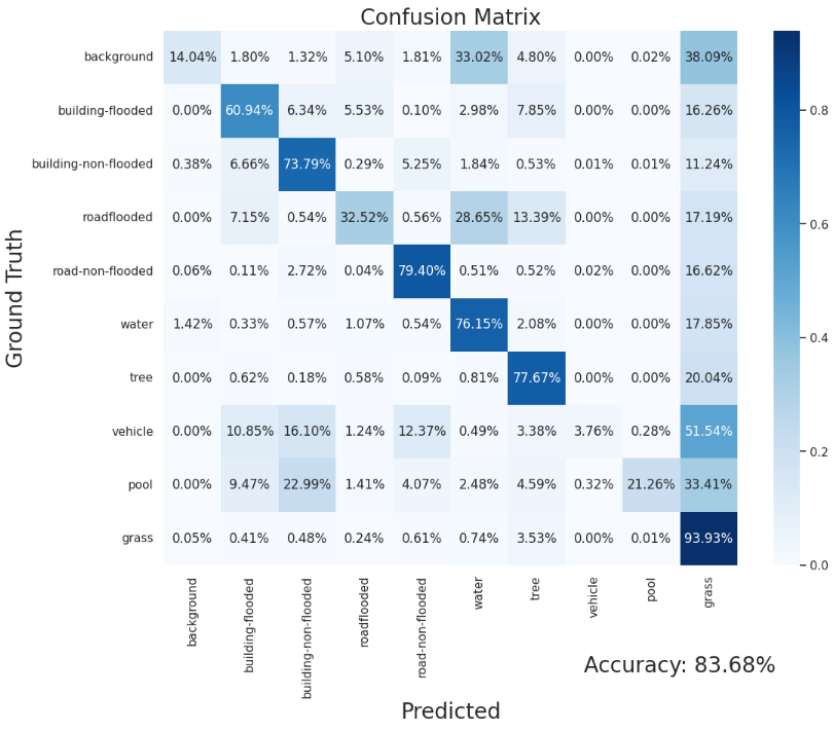
  The trend in mIoU is consistent with the accuracy and loss plots. It shows good initial learning followed by signs of overfitting. The oscillation in validation mIoU suggests inconsistency in model performance on the validation set

3.3.4 Confusion Matrices review

The confusion matrices presented in Figure 3.3 illustrate the performance of both Deeplabv3+ and U-Net across all classes.



(a) Confusion Matrix of DeepLabV3+.



(b) Confusion Matrix of Unet.

Figure 3.3: Confusion Matrices of DeepLabV3+ and Unet.

As previously discussed, there is a significant difference in performance be-

tween Deeplabv3+ and Unet. This disparity is evident in the confusion matrices. Deeplabv3+ demonstrates high accuracy across nearly all classes, achieving 90% for water, 90.6% for grass, 89% for road-non-flooded, and high accuracy for buildings-flooded. However, it struggles with the Vehicle class, achieving only 37.6%. In contrast, Unet faces challenges with certain classes, such as Vehicles 3% and Pool 21.3%. Nevertheless, it performs well in some categories, with 93.9% accuracy for grass, 79.4% for road-non-flooded, and 77.7% for trees.

3.3.5 Visual Results Comparison

Figure 3.4 presents the visual results of comparing U-Net and DeeplabV3+ with the ground truth masks of several samples.

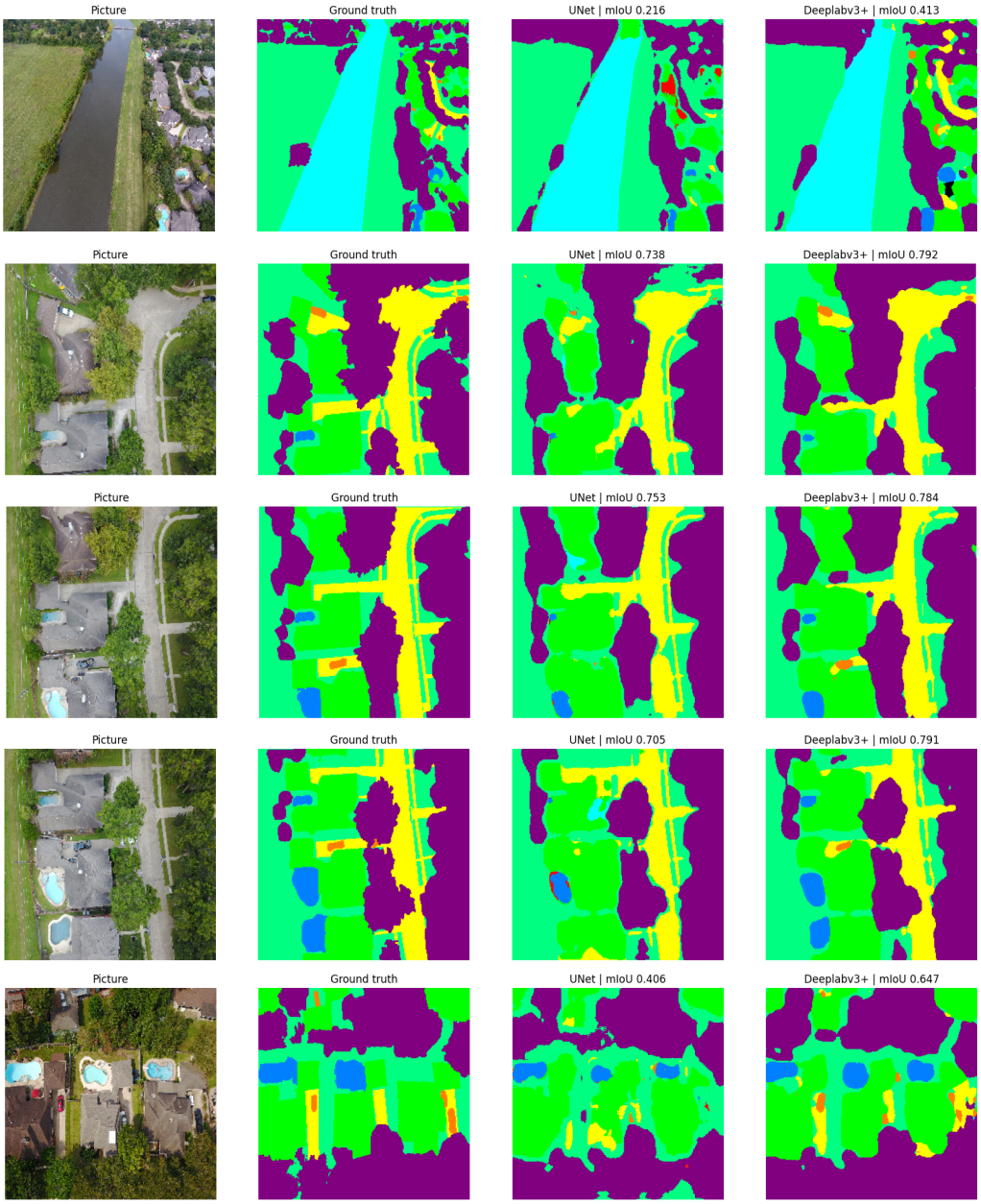


Figure 3.4: Visual results for the semantic segmentation using DeepLabV3+ vs Unet

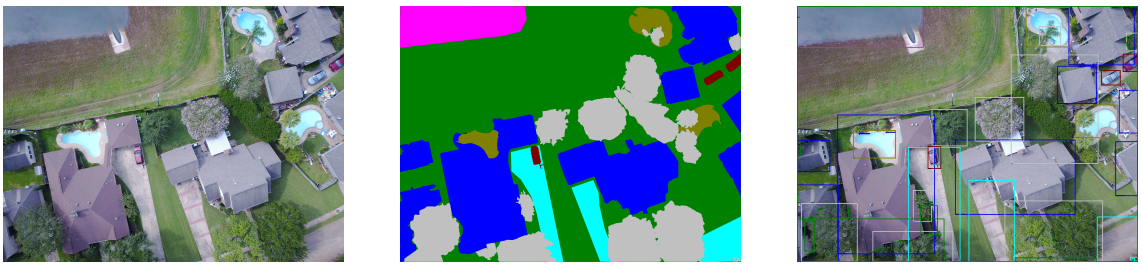
3.4 Object Detection

3.4.1 Dataset Preparation

The initial FloodNet-Supervised v1.0 dataset consisted of high-resolution aerial images annotated with pixel-level segmentation masks, ideal for semantic segmentation tasks. However, YOLO object detection models require bounding boxes instead of detailed segmentation masks. Therefore, we converted these masks into bounding boxes. So to convert these masks into bounding boxes, we follow the steps below:

- **Extraction of Contours:** The first step involved extracting contours from the segmentation masks. These contours are continuous lines that outline the boundaries of objects, identified through changes in image intensity or color.
- **Bounding Box Calculation:** Once the contours were defined, the smallest possible bounding rectangle that could fully enclose each contour was calculated. This rectangle is specified by its top-left corner coordinates (x, y) and its dimensions $(width, height)$, capturing the essential spatial location and size of each object.
- **Normalization of Coordinates:** The bounding box coordinates were then normalized relative to the image dimensions to ensure compatibility with the YOLO model. This involves scaling the coordinates so that the bounding box's center and dimensions are expressed as fractions of the total image width and height.
- **Annotation File Creation:** An annotation file in `.txt` format was created for each image. These files detail the bounding boxes, with each line formatted as `<object-class> <x_center> <y_center> <width> <height>`, suitable for YOLO training requirements.

The restructured FloodNet dataset is now optimally configured for training and deploying YOLO object detection models, which are highly effective for identifying and categorizing objects in high-resolution aerial images of disaster-affected areas. Figure 3.5 shows an example of an image and its mask from the original FloodNet-Supervised v1.0, alongside the annotated image from the new FloodNet dataset.



(a) image (b) original mask (c) annotated image

Figure 3.5: Example of an image, mask, and the annotated image from the new floodnet dataset.

3.4.2 Training Hyperparameters for YOLO

The hyperparametres that we have used in the YOLO training are shown in the table below:

Table 3.3: Training Hyperparameters for YOLO

Hyperparameter	Value	Description
Model Architecture	YOLOv8n	The YOLO version used
Epochs	300	Total number of training epochs
Batch Size	64	Number of training samples per batch
Learning Rate (lr0)	0.00001	Initial learning rate
Optimizer	SGD (auto tuned)	Optimization algorithm used
Image Size (imgsz)	640x640	Dimensions of input images
Workers	16	Number of worker threads for data loading
Training Device	GPU (CUDA)	Hardware used for training
Project	Specified directory	Path for saving training outputs
Exist OK	True	Overwrite existing files without error
Loss Components	Box, Class, DFL	Loss functions used (Box loss, Class loss, DFL)
Momentum	0.937 (auto tuned)	Momentum factor for optimizer
Weight Decay	0.0005	L2 regularization coefficient
Scheduler	Cosine Annealing	Learning rate scheduler used
Pretrained Weights	Yes	Use pretrained model weights

3.4.3 Evaluation and Results

Table 3.4 evaluates the YOLOv8 model's performance on the FloodNet dataset, detailing the total instances for each category and the model's accuracy in identifying true positives. It provides insights into precision using Mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.50 (mAP@0.50) and mAP across a range of 0.50 to 0.95 (mAP@0.50-0.95). Higher values in these metrics indicate better performance in correctly identifying and outlining damaged areas. Additionally, the Recall metric highlights the model's effectiveness in detecting true positives across various damage categories. Here are the detailed results:

Class	Instances	Box(p)	R	MAP50	MAP50-95
All	4821	0.656	0.402	0.544	0.396
Background	207	0.615	0.0773	0.352	0.31
Building-flooded	687	0.665	0.699	0.672	0.545
Building-non-flooded	691	0.667	0.627	0.687	0.537
Road-flooded	326	0.473	0.135	0.296	0.176
Road-non-flooded	717	0.746	0.439	0.622	0.481
Water	1076	0.762	0.205	0.499	0.423
Vehicle	883	0.575	0.424	0.519	0.297
Pool	234	0.747	0.607	0.703	0.401

Table 3.4: Performance Evaluation of the YOLOv8 Model

Figure 3.6 shows the manual labeling details of objects in the dataset through four subfigures:

- **Bar Chart (Top Left)**: Shows the number of instances per category. The category "Water" is particularly prominent, depicted in bright green, showing more instances than other categories.
- **Bounding Box Distribution (Top Right)**: Shows typical object locations within images, using colors to indicate the concentration of bounding boxes.
- **Heatmap of Positional Density (Bottom Left)**: Uses shades of blue to show how often objects appear at different points, with darker colors indicating higher frequencies.
- **Object Dimensions Distribution (Bottom Right)**: Depicts the heights and widths of objects. Lighter shades represent higher frequencies of objects with those dimensions.

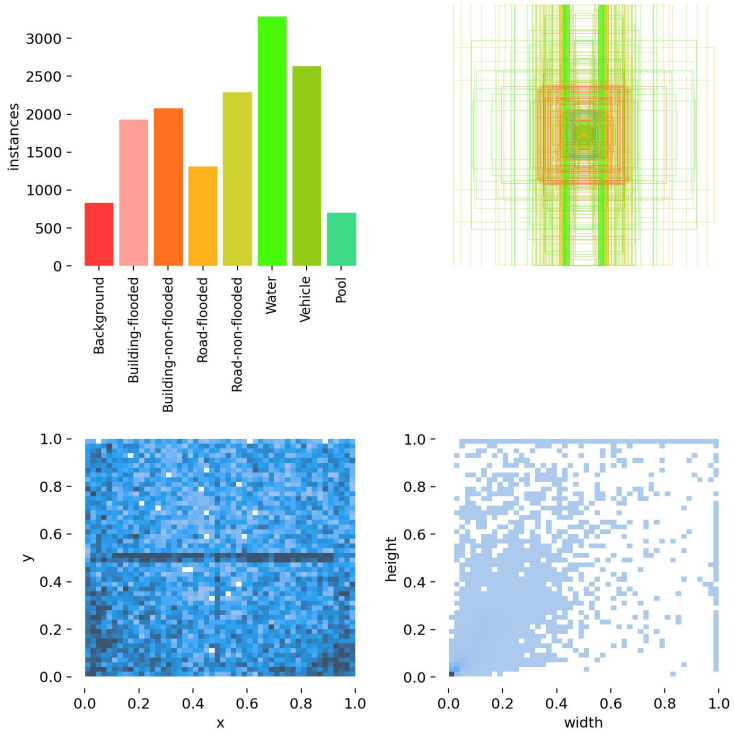


Figure 3.6: Information about objects in FloodNet Track 1 dataset

The Figure 3.7 displays a normalized confusion matrix, a valuable tool used to evaluate the accuracy of an object detection model. This matrix helps us understand how well the model has performed in classifying various types of objects and scenarios within the dataset.

Explanation of the Confusion Matrix:

- **Structure and Function**: The matrix is set up with the actual categories of the objects listed along the bottom (True) and the model's predicted categories along the side (Predicted). Each cell shows what percentage of each

actual category was predicted as each category by the model. The cells along the diagonal line from top left to bottom right show the percentage of correct predictions for each category, which are crucial for assessing the model's accuracy.

- **Visual Guide**: The shading of the cells, from lighter to darker blue, visually represents lower to higher percentages, making it easier to spot which categories the model is most and least accurate at predicting.

This confusion matrix not only quantifies the model's overall performance but also pinpoints specific categories where the model may need further training or adjustment. Particularly, it highlights the need for improved distinction in the model's handling of similar categories, such as differentiating between flooded and non-flooded roads.

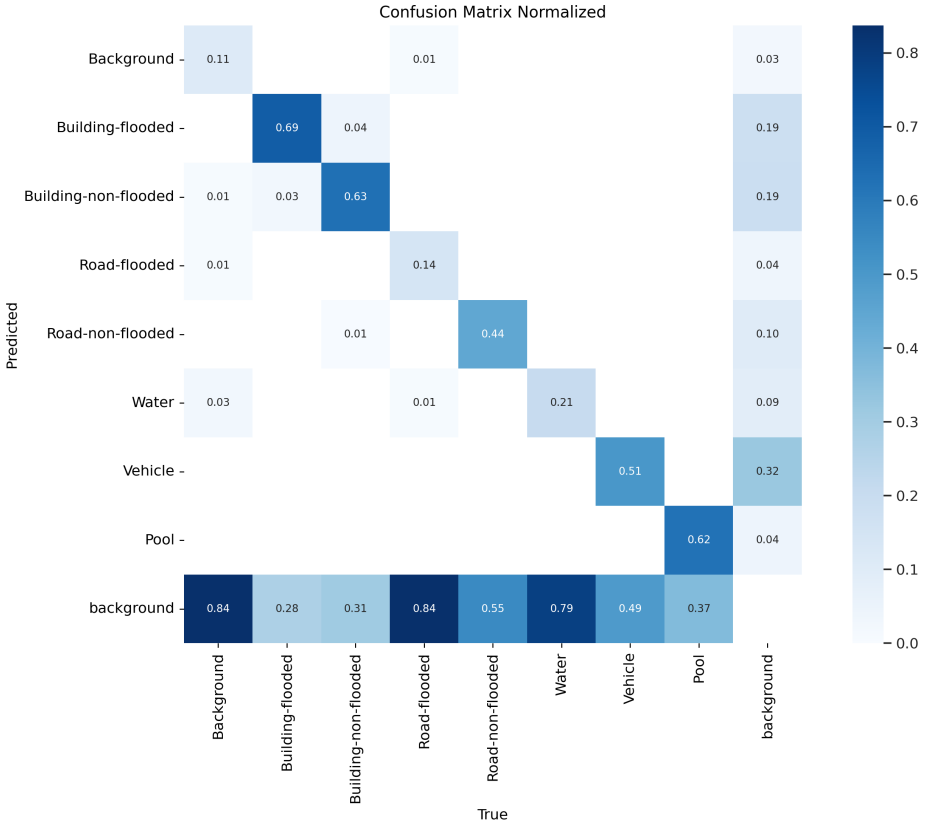


Figure 3.7: The confusion matrix normalized of objects in FloodNet Track 1 dataset

Figure 3.8 showcases a variety of graphs that help us understand how our model is learning and performing through both its training and validation stages.

- **Training Loss (box-loss, cls-loss, dfl-loss)**: These graphs track the loss during training, which is essential for seeing how well the model is picking up on how to predict bounding boxes and classify different objects correctly. As these curves trend downward, it indicates that the model is effectively learning and improving over time.

- **Validation Loss (box-loss, cls-loss, dfl-loss)**: These graphs represent the model's performance on a validation dataset, which includes data that the

model hasn't encountered during its training. A gradual decrease in these graphs is a good sign, showing that the model is performing well and can generalize its learning to new, unseen data.

- **Performance Metrics (precision, recall, MAP50, MAP50-95)**: These curves measure how precise and reliable the model is in detecting objects' locations and how many of the actual objects it can correctly identify. Increasing trends in these metrics are promising, suggesting that the model is becoming more accurate and dependable at pinpointing where objects are in the images.

Together, these graphs give a detailed picture of the model's ongoing progress and effectiveness, highlighting its capabilities and areas where it might still need some improvement. This visual feedback is crucial for tweaking the model to ensure it delivers the best possible performance.

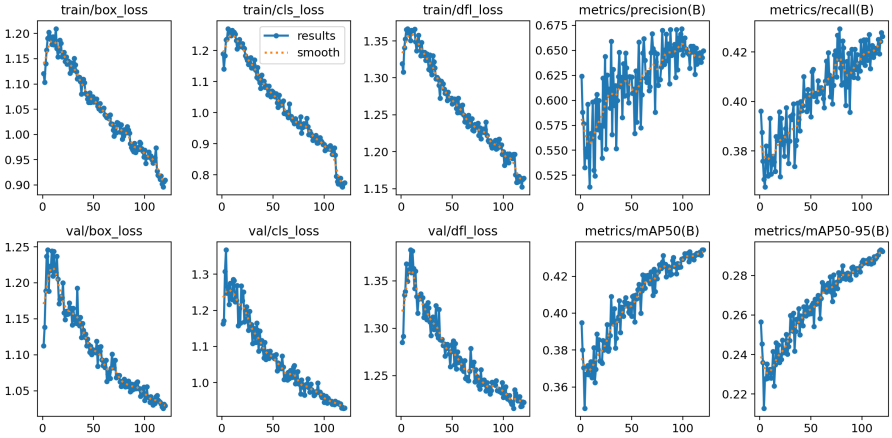


Figure 3.8: Graphs Performance Metrics Curves for the Yolo Model

3.4.4 Visual results

Figures 3.9 and 3.10 display visual results from the application of the YOLOv8 model on the FloodNet dataset.

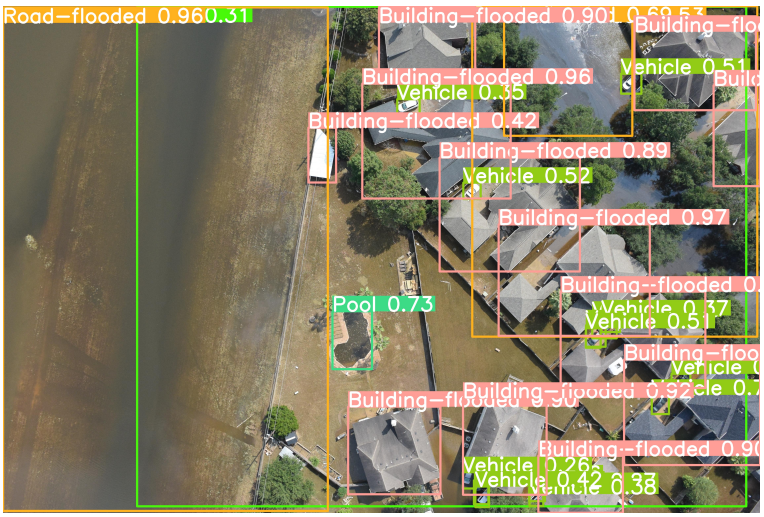


Figure 3.9: Examples 01 of visual results for the Object Detection using Yolov8

Figure 3.10: Examples 02 of visual results for the Object Detection using Yolov8

3.5 Visual Question Answering

3.5.1 FloodNet Track 2 Dataset

The FloodNet Track 2 dataset consists of 2,343 high-resolution aerial photographs captured using DJI Mavic Pro quadcopters following Hurricane Harvey. This dataset is designed to support disaster response enhancements through a Visual Question Answering (VQA) framework.

3.5.1.1 Dataset Composition

The FloodNet Track 2 dataset is divided into three primary sets, as detailed in the table below:

Set	Number of Images	VQA Pairs
Training	1450	4511
Validation	468	1415
Test	425	1429
All	2343	7355

Table 3.5: Composition of the FloodNet Track 2 Dataset

3.5.1.2 Annotations and Question Types

The dataset provides detailed semantic annotations that support various types of VQA tasks. The types of questions included in the VQA tasks are summarized in the following table:

Question Type	N. Training	N. Validation	N. Testing	All
Simple Counting	636	197	201	1034
Complex Counting	693	216	222	1131
Condition Recognition	2315	726	728	3769
Yes/No Questions	867	276	278	1421

Table 3.6: Statistics of Question Types in the FloodNet Track 2 Dataset

Question Type	Description	Possible Answer
Simple Counting	Questions such as "How many buildings are in the image? ", "How many buildings are in this image?", "How many buildings can be seen in the image?", "How many buildings can be seen in this image?"	1,2,3,4...
Complex Counting	Questions require counting based on attributes, e.g., "How many buildings are flooded in this image?", "How many buildings are flooded?", "How many buildings are non flooded in this image?", "How many buildings are non flooded?", "How many flooded buildings can be seen in this image?", "How many non flooded buildings can be seen in this image?"	1,2,3,4...
Condition Recognition	Questions focus on the condition of infrastructure, e.g., "What is the condition of the road in the given image?", "What is the condition of the road in this image?", "What is the condition of road?"	Flooded, Non-Flooded, Flooded and Non-Flooded
Yes/No Questions	Binary questions such as "Is the entire road flooded?", "Is the entire road non flooded?"	Yes, No

Table 3.7: Types of VQA Tasks in the FloodNet Track 2 Dataset

3.5.1.3 Example of Training Set Usage

This section presents an actual example from the training set of the FloodNet Track 2 dataset. Figure 3.11 shows an image, followed by Table 3.8 , which illustrates a sample of the questions posed and the correct answers based on the provided annotations.

Figure 3.11: Image from the FloodNet Track 2 Training Set

Question Type	Question	Answer
Simple Counting	How many buildings are in this image?	3
Complex Counting	How many buildings are non flooded?	3
Condition Recognition	What is the condition of the road in this image?	non flooded
Yes/No Questions	Is there any flooded road in the image?	No

Table 3.8: Questions and Answers for the Image

3.5.2 Process Flow of the VQA Model

The VQA model integrates image and text processing to generate answers based on visual and textual inputs. Figure 3.12 outlines the process flow, including tokenization with BERT, feature extraction with YOLO, and answer generation with GPT-2.

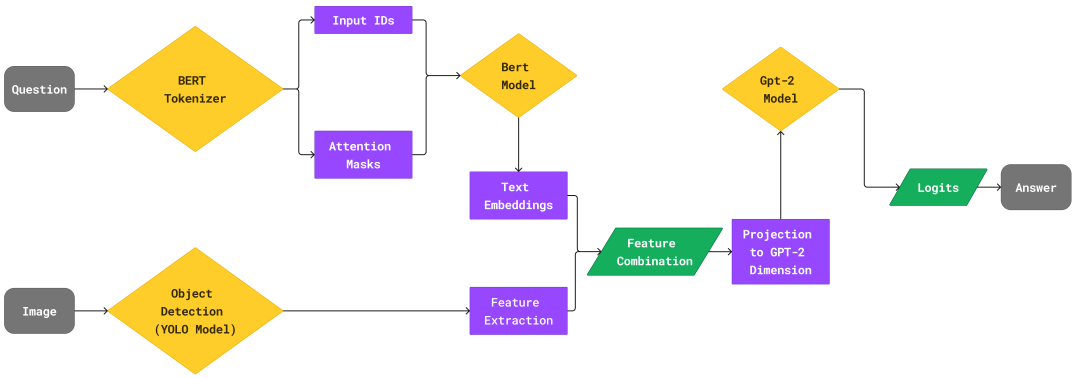


Figure 3.12: The Detailed Flowchart Representing The VQA Model Process

- **Question Input**: The user provides a question related to the image. This text input serves as the query that the VQA model will process to generate an answer.
- **BERT Tokenization**: The question text is tokenized into input IDs and attention masks.
  - **Input IDs**: These are numerical representations of the tokens in the question.
  - **Attention Masks**: Binary masks indicating the presence of actual tokens (1) or padding (0).
- **BERT Embedding**:
  - **BERT Model**: The input IDs and attention masks are passed through the BERT model to obtain embeddings.
  - **Text Embeddings**: High-dimensional vector representations of the question, specifically using the pooler output for the [CLS] token.
- **Image Input**: The user also provides an image related to the question. This visual input is processed to extract relevant features.
- **YOLO Feature Extraction**:
  - **Object Detection (YOLOv8)**: The image is processed using the YOLO model to detect objects and extract features.
  - **Image Features**: These include bounding box coordinates, confidence scores, and class values for detected objects.
- **Feature Combination**: The text embeddings obtained from BERT and the image features extracted by YOLO are combined. This step creates a unified representation that integrates information from both text and image inputs.
- **Projection to GPT-2 Dimension**:
  - **Linear Layer**: The combined features are projected to match the input dimension required by GPT-2.

  - **Projected Features**: The transformed combined features are now suitable for processing by the GPT-2 model.

- **Answer Generation with GPT-2**:
  - **GPT-2 Model**: The projected features are passed through the GPT-2 model to generate an answer.
  - **Logits**: The output from GPT-2, which includes the probability distributions used for predicting the answer.

3.5.3 Training Hyperparameters for VQA Model

The hyperparametres that we have used in the VQA training are shown in the table below:

Table 3.9: Training Hyperparameters for VQAModel

Hyperparameter	Value
Learning Rate	0.0001
Batch Size	64
Number of Epochs	100
Optimizer	Adam
Loss Function	CrossEntropyLoss
Hidden Dimension	256
Max Length for Tokenizer	64
Image Transformations	Resize (224, 224), ToTensor()
Device	GPU (CUDA)

3.5.4 Evaluation and Results

3.5.4.1 Training Performance Metrics

This section presents the training performance metrics for the VQA model. The figures 3.13 , 3.14 illustrate the average accuracy and average loss per epoch for 100 training epochs.

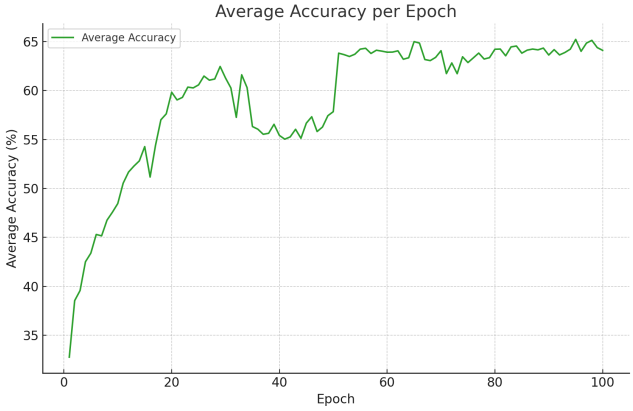


Figure 3.13: Average Accuracy VQA Model

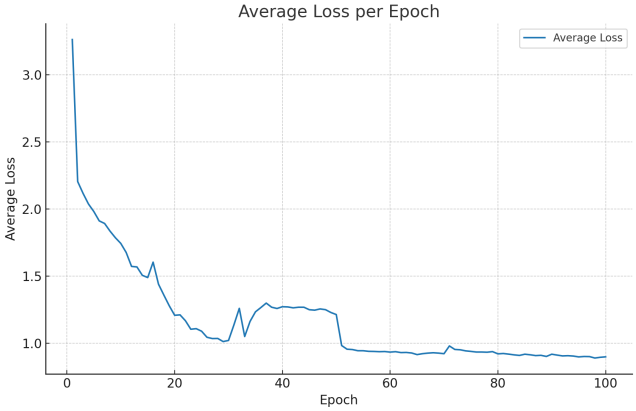


Figure 3.14: Average Loss VQA Model

The figure 3.13 displays the average accuracy achieved by the VQA model at each training epoch. The accuracy is measured as the proportion of correct answers out of the total questions answered. The graph demonstrates the model's learning progress and convergence over time.

- The accuracy increases significantly during the initial epochs, indicating rapid learning.
- There are fluctuations in accuracy, which are common in training deep learning models due to the stochastic nature of optimization algorithms.
- After approximately 50 epochs, the accuracy stabilizes around 0.65, indicating that the model has reached a point where additional training epochs yield diminishing improvements.

The figure 3.14 illustrates the average loss per epoch during the training process. The loss function quantifies the difference between the model's predictions and the actual answers, with lower values indicating better model performance. The graph provides insight into the model's convergence behavior.

- The average loss decreases sharply in the initial epochs, reflecting the model's rapid adaptation to the training data.
- There are occasional increases in loss, which can occur due to the learning rate adjustments.
- After around 60 epochs, the loss stabilizes just below 1.0, suggesting that the model has reached a state of consistent performance with minimal further improvement.

3.5.4.2 Validation Performance Metrics

The validation of the VQA model provided the following results:

Metric	Accuracy (%)	Average Loss
Overall	64.51	0.8833
Condition Recognition	91.02	0.2521
Yes/No Questions	51.56	0.7046
Simple Counting	27.67	2.0694
Complex Counting	25.97	2.1270

Table 3.10: Validation Performance Metrics

During validation, the model's performance was evaluated across different question types. The results highlight significant variation in accuracy and loss among these types, indicating areas where the model performs well and others where improvements are needed. The overall accuracy of the model was 64.51%, with an average loss of 0.8833. The model performed exceptionally well in *Condition Recognition* tasks, achieving an accuracy of 91.02% with a low average loss of 0.2521. For *Yes/No Questions*, the model achieved a moderate accuracy of 51.56% and an average loss of 0.7046. However, the model struggled with *Simple Counting* and *Complex Counting* tasks, with accuracies of 27.67% and 25.97% respectively, and higher average losses of 2.0694 and 2.1270.

3.5.4.3 Model Predictions

This section provides examples of the VQA model's performance on various inputs. Each example includes the input image, the question posed to the model, and the answer generated by the model.

Figure 3.15: Example 1 of the VQA model predictions

Figure 3.16: Example 2 of the VQA model predictions

Example	Question	Model's Answer
Example 1	How many buildings are in this image?	13
Example 1	How many buildings are flooded in this image?	13
Example 1	Is the entire road non flooded?	No
Example 1	What is the condition of the road in this image?	Flooded
Example 2	How many buildings are in this image?	4
Example 2	How many buildings are non flooded in this image?	3
Example 2	Is the entire road non flooded?	No
Example 2	What is the condition of the road in this image?	Non Flooded

Table 3.11: Examples of VQA Model's Performance

In Figure 3.15, the model accurately identifies the condition of the road as flooded. Similarly, in Figure 3.16 the road as non flooded, the model correctly counts the number of visible buildings. These examples demonstrate the model's ability to process both condition recognition and counting tasks effectively. Additional questions and their corresponding answers are provided in Table 3.11.

3.6 Deep learning Software and Tools

In this section, we will define and explain the programming languages, software, and tools utilized in the development of our application.

3.6.1 Python programming language

Python, created by *Guido van Rossum*, is a high-level programming language known for its readability and simplicity. It uses indentation to define code blocks, enhancing legibility and easing maintenance. Supporting procedural, object-oriented, and

functional programming, Python is versatile for web development, scientific computing, and AI. Its extensive standard library and interpretive nature enable quick development. As an open-source language, Python promotes collaborative development and is widely used across various fields [35].

3.6.2 Google Colaboratory

Google Colaboratory *Colab* is a cloud platform using Jupyter Notebook technology for machine learning and deep learning research. It integrates with Google Drive for easy sharing and collaboration. Colab provides a free, powerful GPU for intensive tasks, especially with TensorFlow and Pytorch, though it has CPU and operation time limitations. Despite this, Colab is a valuable, maintenance-free tool for education and preliminary AI studies [36].

3.6.3 Deep Learning Framework and Library

3.6.3.1 PyTorch

PyTorch is an open-source machine learning and deep learning library developed by Facebook, Inc. It is Python-based, as its name suggests, and aims to provide a faster alternative/replacement to NumPy) by providing a seamless use of GPUs and a platform for deep learning that provides maximum flexibility and speed [37].

3.6.3.2 Ultralytics

Ultralytics offers a comprehensive suite of tools and libraries designed for the field of artificial intelligence and machine learning. It is specifically tailored to streamline the development and training of deep neural network models, with a particular focus on image analysis and object detection. The platform provides an array of powerful features and user-friendly APIs, enabling researchers and practitioners to efficiently conduct experiments and thoroughly analyze their results.

3.7 Conclusion

In this chapter, we have explored various deep learning approaches to address critical tasks in disaster management, specifically focusing on semantic segmentation, object detection, and visual question answering using the FloodNet dataset. The detailed experiments and evaluations highlight the efficacy of advanced models such as DeepLabV3+ and YOLOv8 in accurately segmenting and detecting objects in flood-affected areas. The VQA model demonstrated its capability to integrate textual and visual information, providing insightful answers to questions about the images. Despite the promising results, the varying performance across different tasks indicates areas for further improvement, particularly in complex counting scenarios. The insights gained from these experiments underscore the potential of deep learning technologies in enhancing disaster response and management strategies.

Chapter 4

DisasterVision System Web Application

4.1 Introduction

In this chapter we present our web application that leverages cutting-edge computer vision techniques, including semantic segmentation, object detection, and visual question answering (VQA), to enhance disaster response capabilities. By integrating these technologies, the application not only automates identifying and categorizing disaster-related objects and regions in aerial images but also allows users to interactively query the system for specific information about the disaster scene. This holistic approach facilitates real-time analysis and decision-making, providing emergency responders and decision-makers with critical insights that can save lives and resources.

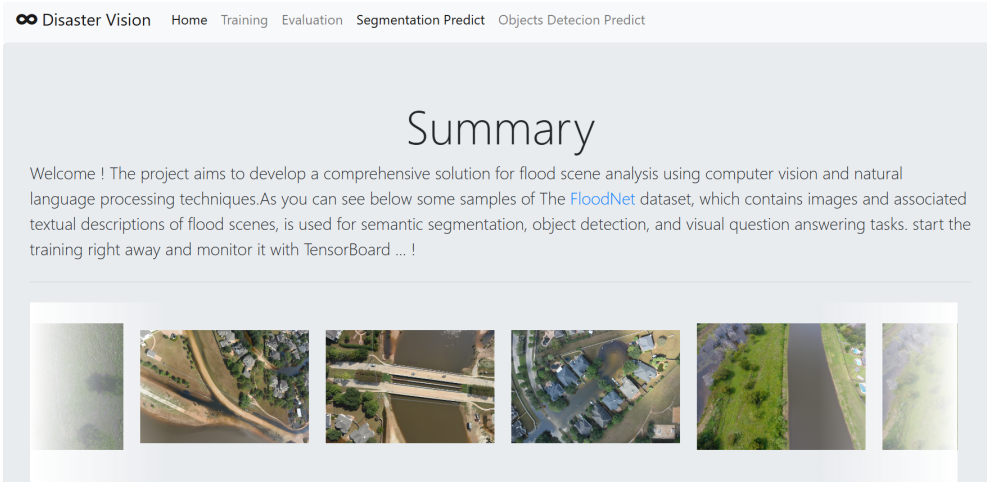
The application encompasses three primary functionalities: model training, prediction, and evaluation. It harnesses large datasets of annotated aerial imagery through model training to build robust and accurate models. The prediction module applies these trained models to new, unseen data, precisely identifying and segmenting disaster-affected areas. Finally, the evaluation component ensures continuous improvement and validation of the models by assessing their performance against ground truth data.

4.2 Disaster Vision System Graphical User interfaces

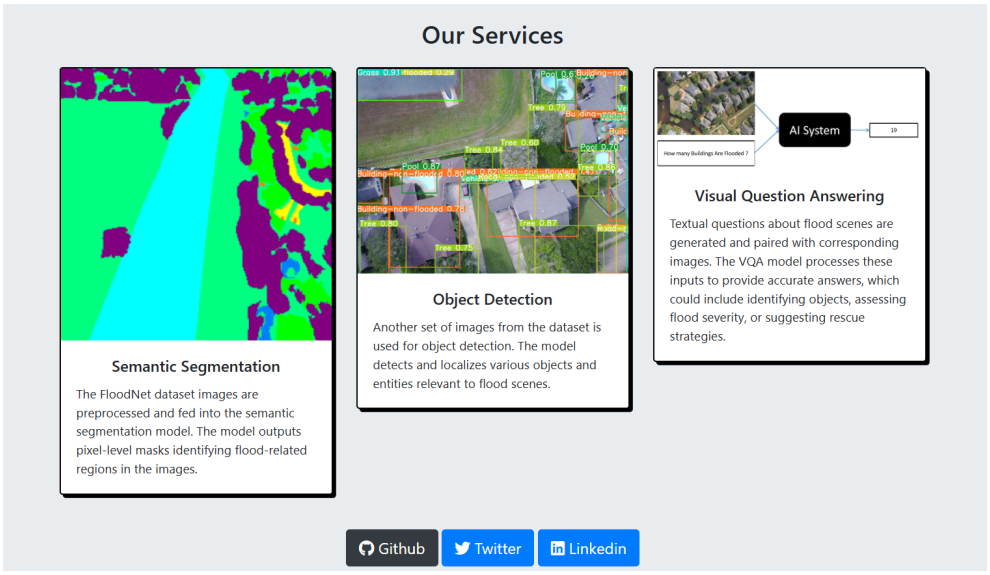
Here we present for you our application view windows with a description of its functionalities

4.2.1 Main View

This view shown in Figure 4.1 provides an overview of the Disaster Vision project, which focuses on flood scene analysis using computer vision and natural language processing techniques. It showcases samples from the FloodNet dataset and emphasizes its applications in semantic segmentation, object detection, and visual question answering.



(a) Summery and Dataset Samples



(b) DisasterVision Web App Services.

Figure 4.1: Main View - Index

Followed by a description of the three main services offered by the application which are training, evaluation, and prediction offering models of various approaches like segmentation, object detection, and visual question answering.

4.2.2 Training View

This view shown in Figure 4.2 , the user can choose one of the three Deep-learning approaches to train a model

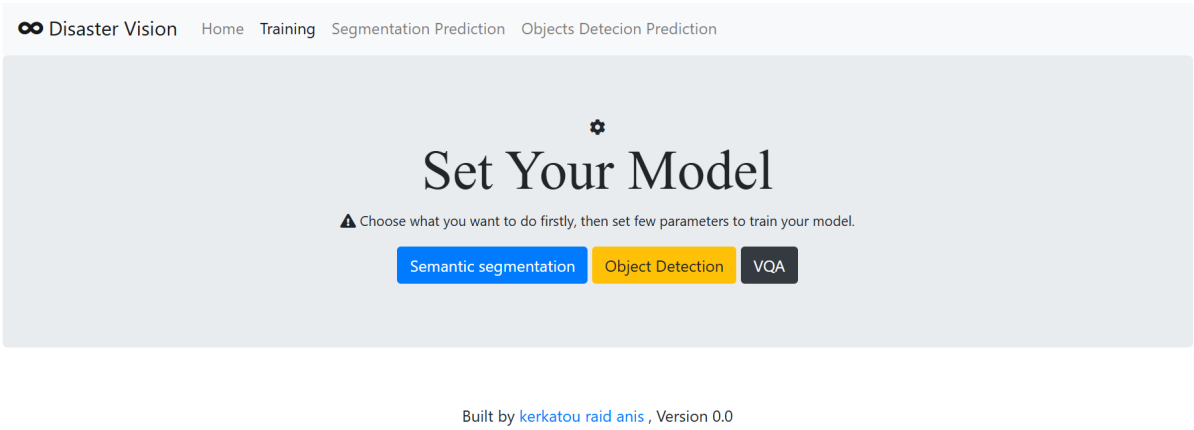


Figure 4.2: Training Main View

4.2.2.1 Semantic Segmentation Training

This view shown in Figure 4.3 is used for setting up the training configuration. Users can specify parameters (as shown in Figure 4.4) such as dataset path, type of Device, pre-trained models, device type, number of classes, batch size, epochs, learning rate, and image size. The "Start Training!" button initiates the training process.

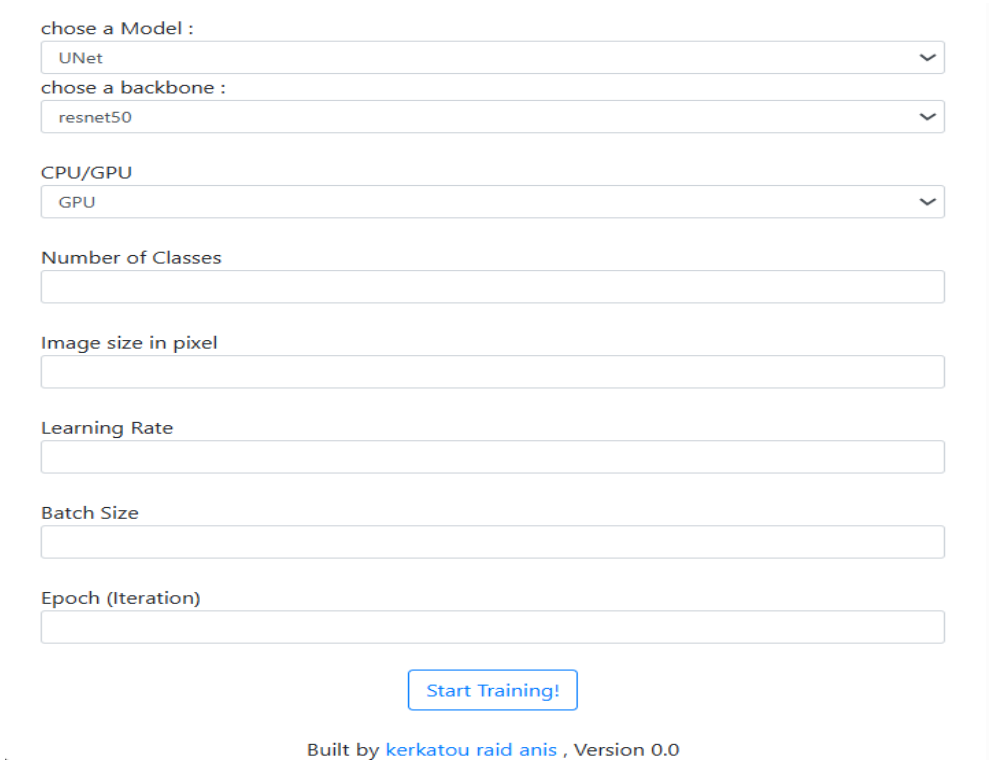


Figure 4.3: Semantic Segmentation Training View

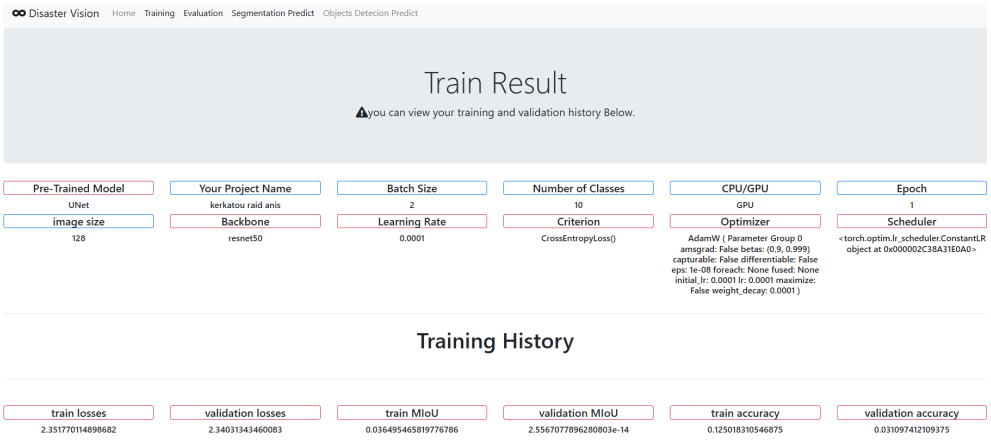


Figure 4.4: Training History

The training results provide an overview of the training history, encompassing details such as learning rate, image size, and optimizer configuration. The subsequent section presents the model's performance metrics, including training loss, mean Intersection over Union (mIoU), accuracy values, and validation too.

4.2.2.2 Object Detection Training

This view is used for setting up the training configuration. Users can specify parameters (as shown in Figure 4.5) such as dataset path, type of label, pre-trained models, device type, number of classes, batch size, epochs, learning rate, number of workers, and image size. The "Start Training!" button initiates the training process

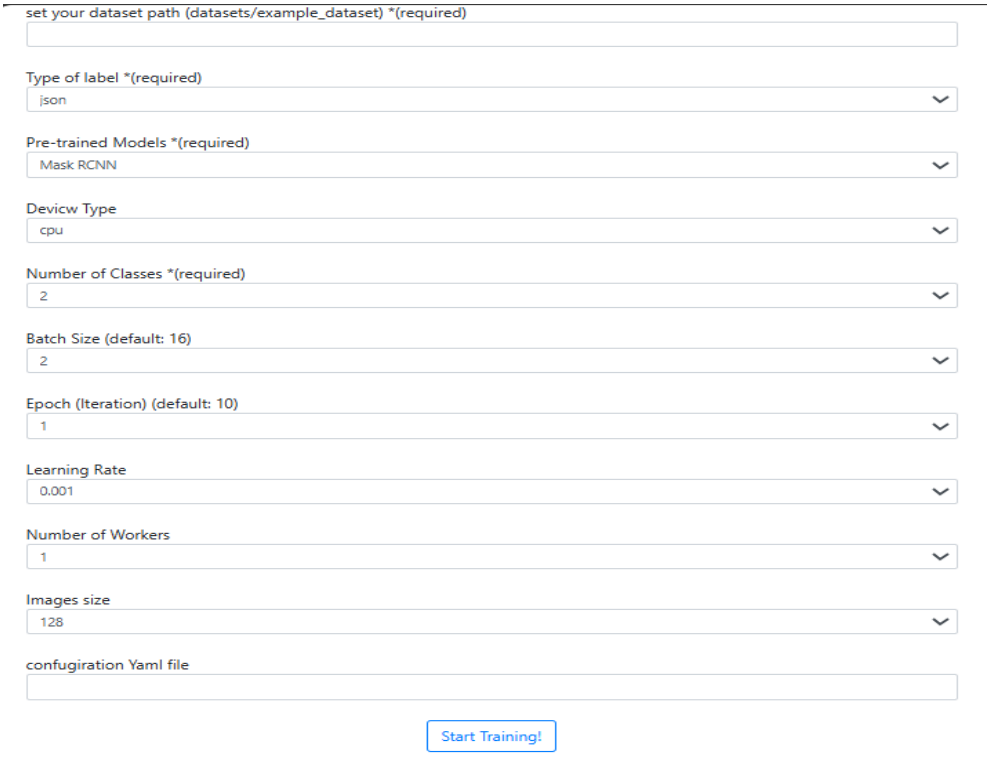


Figure 4.5: Object Detection Training View

Figure 4.6: Training History

4.2.2.3 Visual Question Answering Training

As well as this view that is used for setting up the training configuration of Visual Question Answering models. Users can specify parameters (as shown in Figure 4.7) such as images and Questions dataset paths, pre-trained models, device type, number of classes, batch size, number of epochs, and learning rate. The "Start Training!" button initiates the training process

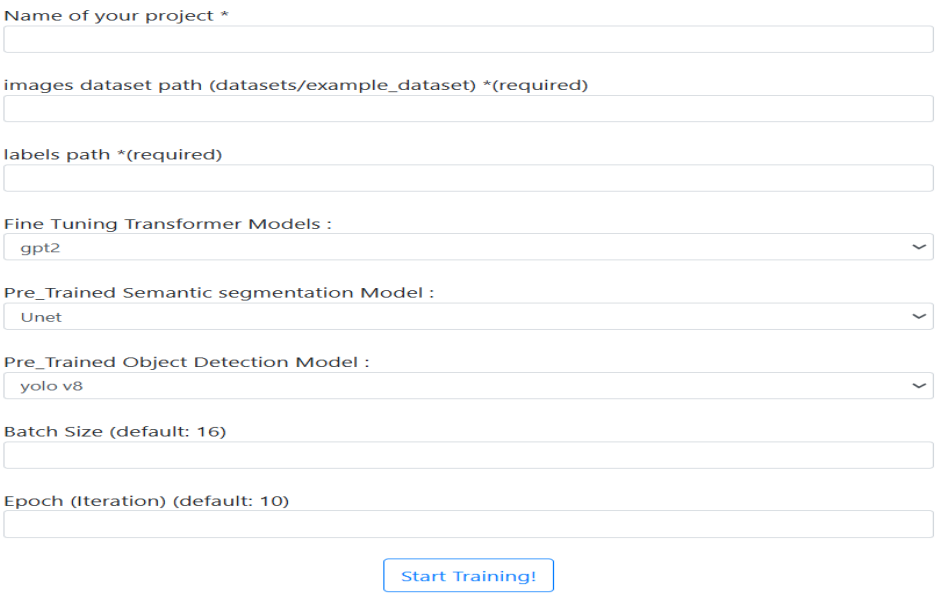


Figure 4.7: Visual Question Answering Training View

Upon completion of the training, the following view (as shown in Figure 4.8) will appear to inform the user that the training has successfully concluded.

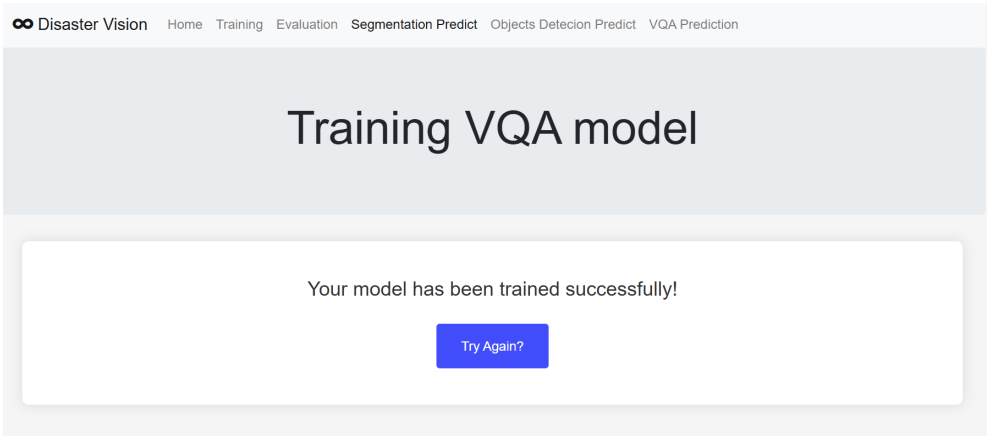
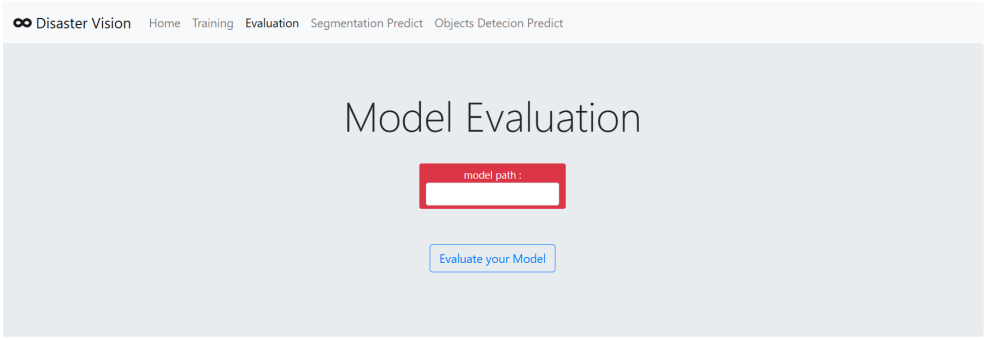


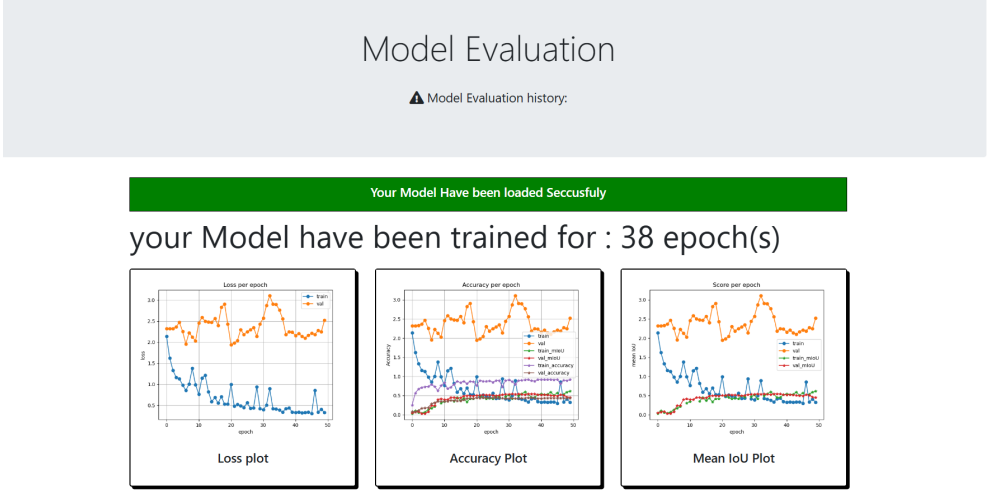
Figure 4.8: Visual Question Answering Training Result View

4.2.3 Evaluation View

In this view shown in Figure 4.9 you can load your semantic segmentation model to view its training information by clicking on **evaluate your model**, including loss, accuracy, and IoU plots, as well as the number of epochs it has been trained and the confusion matrix of the classes.



(a) Model Evaluation



(b) Model Evaluation Results

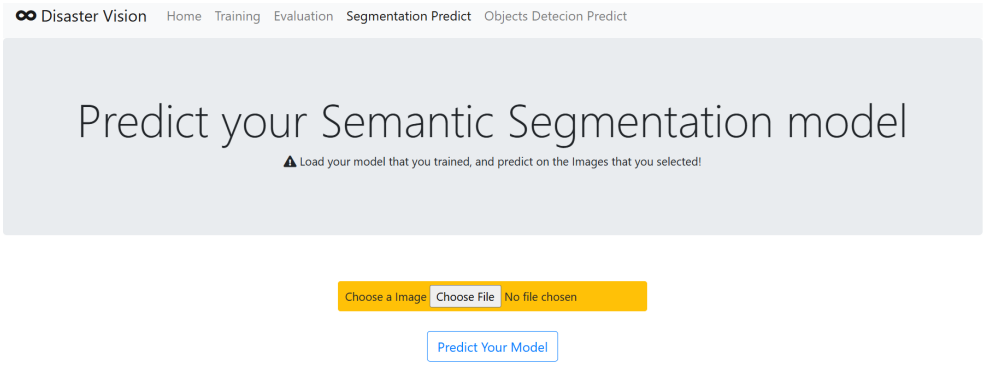
Figure 4.9: Evaluation Views

4.2.4 Prediction Views

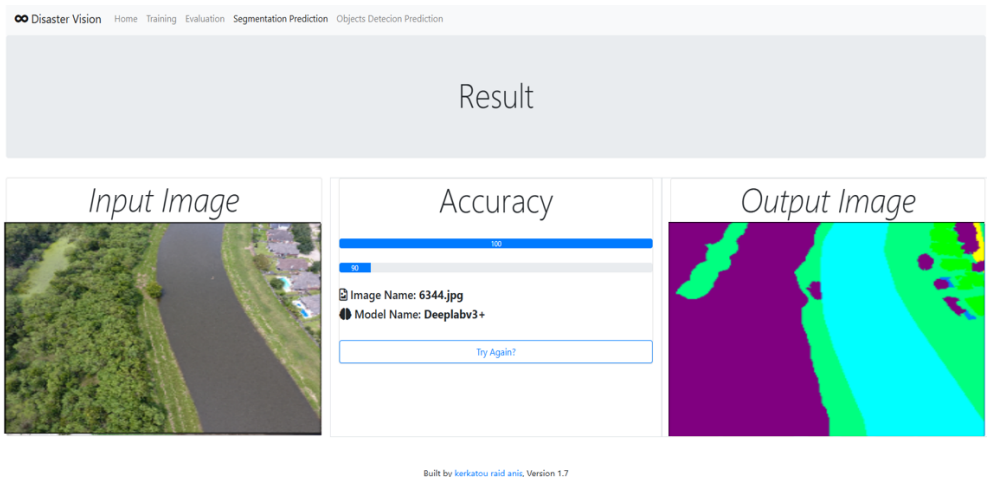
In this part, we can see three types of prediction, in Semantic Segmentation you need to choose an image and then press the prediction button, in the case of visual question answering you need to choose an image and a question.

4.2.4.1 Semantic Segmentation Prediction

For Semantic Segmentation, you can select any image and then click on **Predict Your Model**. This will automatically redirect you to the results view, where you can view the segmented image with color-coded regions as shown in Figure 4.10 .



(a) Semantic Segmentation Prediction

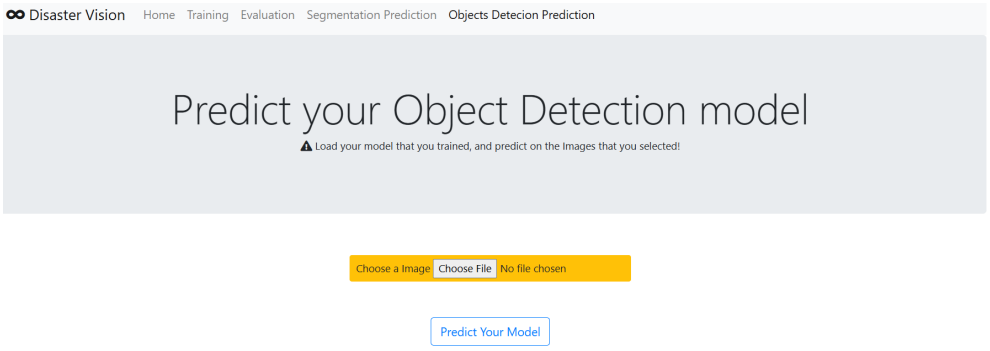


(b) Semantic Segmentation result

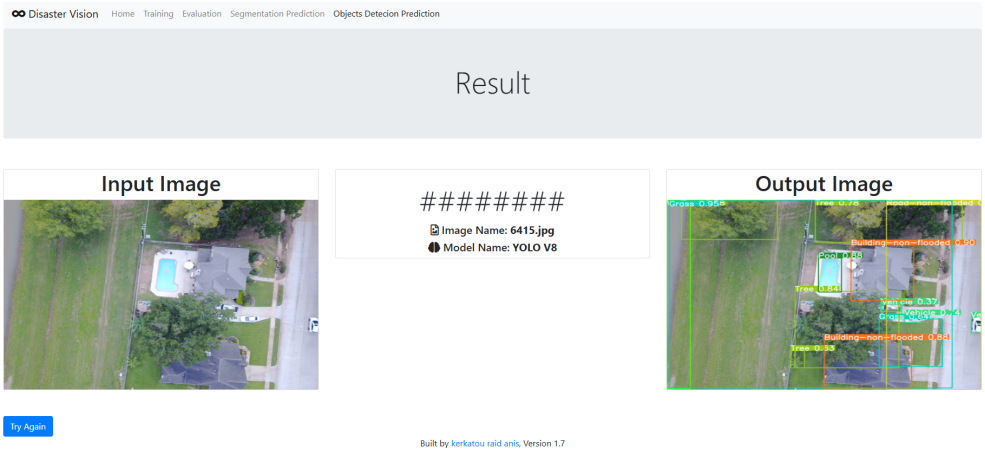
Figure 4.10: Prediction and Result Views

4.2.4.2 Object Detection Prediction for Images and videos

For Object Detection, you can choose between an image or a video and then click on **Predict Your Model** as shown in Figure 4.11 . This will automatically redirect you to the results page, where you can view the detected objects within the selected image or video.



(a) Objects Detection Prediction



(b) Object Detection Result

Figure 4.11: Object Detection Prediction and Result Views

4.2.4.3 Visual Question Answering Prediction

Finally, for Visual Question Answering, you can select any image and enter a question, then click on **Generate Answer** as shown in Figure 4.12 . This will automatically redirect you to the results view as shown in Figure 4.13 , where the answer to your question will be displayed below the image and the question.

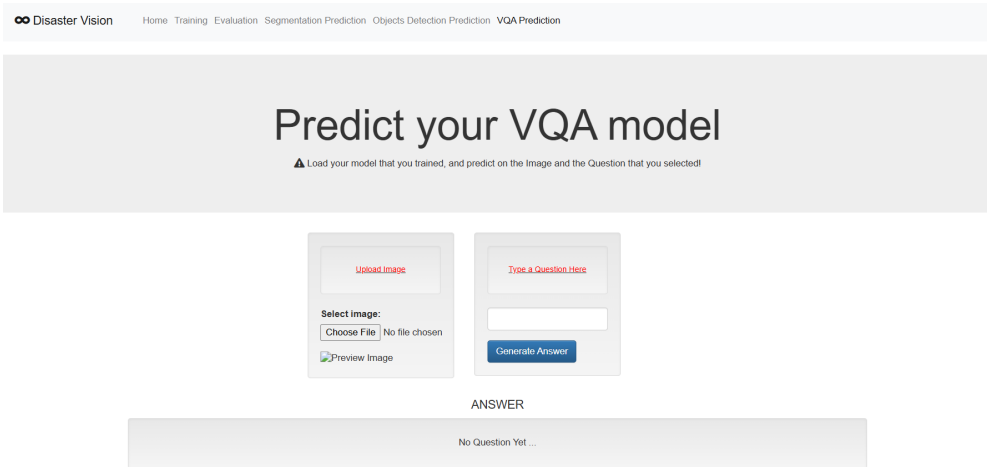


Figure 4.12: Visual Question Answering Prediction

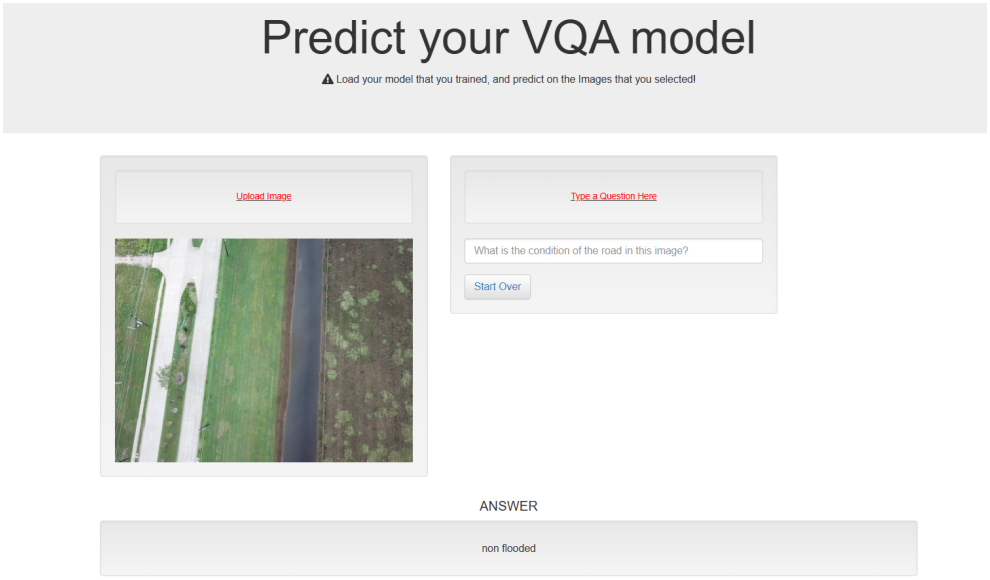


Figure 4.13: Visual Question Answering result

4.2.5 System Deployment Tools and Frameworks

4.2.5.1 Web Development Tools

HTML is used to structure the content of a webpage, defining elements like headings, paragraphs, and images. CSS is used for styling and layout, allowing you to control the appearance of the HTML elements, such as colors, fonts, and spacing. JavaScript is a programming language that enables interactive elements on web pages.

4.2.5.2 Bootstrap Framework

Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites.

4.2.5.3 Flask Framework

Flask is a lightweight web framework for Python that helps you build web applications easily and quickly. It provides the tools and features needed to create a web server, handle requests, and generate responses with minimal code.

4.2.5.4 Ajax and jQuery

AJAX (Asynchronous JavaScript and XML) with jQuery is a web development technique that allows parts of a web page to be updated without reloading the entire page. It uses JavaScript to send and receive data asynchronously with a server. jQuery simplifies the use of AJAX by providing convenient methods to perform these requests.

4.3 Conclusion

The development and deployment of this web application represent a significant step forward in utilizing computer vision for disaster management. By combining semantic segmentation, object detection, and visual question answering, the application provides a comprehensive tool for analyzing aerial imagery in the aftermath of disasters. This integration not only streamlines the assessment process but also enhances the accuracy and speed of response, ultimately contributing to more effective disaster mitigation and recovery efforts.

As the application evolves, further enhancements and refinements will be guided by ongoing feedback from users and advancements in the underlying technologies. The potential for integrating additional data sources, improving model accuracy, and expanding the range of disasters covered is vast. By continuing to leverage the power of computer vision, this application aims to be at the forefront of innovative solutions for disaster management, offering a critical resource in the face of increasing global challenges posed by natural disasters.

General conclusion

In conclusion, this research aimed to effectively assess damage to buildings and evaluate post-natural disaster damage using aerial image analysis. The primary goal was to create an automated system capable of detecting and categorizing different types of damage, with a particular focus on identifying affected areas. Significant advancements were made in semantic segmentation, object detection, and even natural language processing, thanks to unmanned aerial vehicles (UAVs) and deep learning techniques, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Various models, such as Unet, Deeplabv3, YOLOv8n, BERT, and GPT2, were assessed in this study. Each model exhibited strengths and weaknesses, and their performance was evaluated to achieve precise and reliable results in detecting and classifying different types of damage.

The research findings illustrate the potential of using aerial image analysis and deep learning Approaches to assess post-natural disaster damage efficiently. The developed system offers valuable insights into the extent and types of damage, aiding in rapid response and recovery efforts. However, further research and enhancements are necessary to improve the system's performance and broaden its real-world applications.

By utilizing drones and deep learning algorithms, this research contributes to disaster management, paving the way for more effective and automated assessment methods. The adoption of such technologies can significantly mitigate the impacts of natural disasters and streamline timely responses and recovery operations. Despite constraints in resources and time, this study represents a crucial step toward more efficient disaster relief efforts.

Bibliography

[1] Riskyana Dewi Intan Puspitasari, Fadhilah Qalbi Annisa, and Danang Ariyanto. "Flooded Area Segmentation on Remote Sensing Image from Unmanned Aerial Vehicles (UAV) using DeepLabV3 and EfficientNet-B4 Model". In: *2023 International Conference on Computer Control Informatics and its Applications (IC3INA)*. IEEE. 2023, pp. 216–218. DOI: 10.1109/IC3INA60834.2023.10285752.

[2] F Rahman. "Save the world versus man-made disaster: A cultural perspective". In: *IOP Conference Series: Earth and Environmental Science*. Vol. 235. IOP Publishing. English Department, Hasanuddin University, Indonesia, 2019, p. 012071. DOI: 10.1088/1755-1315/235/1/012071. URL: https://doi.org/10.1088/1755-1315/235/1/012071.

[3] Ruiling Sun et al. "A review of risk analysis methods for natural disasters". In: *Natural Hazards* DOI 10.1007/s11069-019-03826-7 (2019). DOI: 10.1007/s11069-019-03826-7. URL: https://doi.org/10.1007/s11069-019-03826-7.

[4] Jonathan E. Suk et al. "Natural disasters and infectious disease in Europe: a literature review to identify cascading risk pathways". In: *European Journal of Public Health* 30.5 (2019), pp. 928–935. DOI: 10.1093/eurpub/ckz111. URL: https://academic.oup.com/eurpub/article/30/5/928/5512023.

[5] BBC News Kirstie Brewer Paul Kerley. *Turkey earthquake: The eyewitnesses who captured the quake on social media*. Accessed: 2024-05-16. 2023. URL: https://www.bbc.com/news/world-64541194.

[6] Fox Weather. *Mid-Atlantic Fire Outbreak*. Accessed: 2024-05-27. 2024. URL: https://www.foxweather.com/extreme-weather/mid-atlantic-fire-outbreak.

[7] CNN PHOTOGRAPH BY LEANDRO LOZADA / AFP VIA GETTY IMAGES. *A neighborhood in Jackson, Kentucky, is overwhelmed by flash flooding after heavy rains caused the Kentucky River to overflow in July 2022*. Accessed: 2024-05-16. 2022. URL: https://kids.nationalgeographic.com/science/article/flood.

[8] Ashfaq Ahmad Shah et al. "Identifying obstacles encountered at different stages of the disaster management cycle (DMC) and its implications for rural flooding in Pakistan". In: *Frontiers in Environmental Science* 11 (2023), p. 1088126. DOI: 10.3389/fenvs.2023.1088126. URL: https://www.frontiersin.org/articles/10.3389/fenvs.2023.1088126/full.

[9] Nader Mohamed et al. "Unmanned aerial vehicles applications in future smart cities". In: *Technological Forecasting & Social Change* (2018). DOI: 10.1016/j.techfore.2018.05.004. URL: https://doi.org/10.1016/j.techfore.2018.05.004.

[10] Emilien Alvarez-Vanhard, Thomas Corpetti, and Thomas Houet. "UAV & satellite synergies for optical remote sensing applications: A literature review". In: *Science of Remote Sensing* 3 (2021), p. 100019. DOI: 10.1016/j.srs.2021.100019. URL: https://doi.org/10.1016/j.srs.2021.100019.

[11] Tashnim Chowdhury et al. "Comprehensive Semantic Segmentation on High Resolution UAV Imagery for Natural Disaster Damage Assessment". In: *2020 IEEE International Conference on Big Data (Big Data)*. IEEE. 2020, pp. 3904–3905. DOI: 10.1109/BigData50022.2020.9377916.

[12] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[13] Nahian Siddique et al. "U-net and its variants for medical image segmentation: A review of theory and applications". In: *Ieee Access* 9 (2021), pp. 82031–82057.

[14] Liang-Chieh Chen et al. "Encoder-decoder with atrous separable convolution for semantic image segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 801–818.

[15] Huixuan Fu et al. "Bridge Crack Semantic Segmentation Based on Improved Deeplabv3+". In: *Journal of Marine Science and Engineering* 9.6 (2021). ISSN: 2077-1312. DOI: 10.3390/jmse9060671. URL: https://www.mdpi.com/2077-1312/9/6/671.

[16] Md Atiqur Rahman and Yang Wang. "Optimizing intersection-over-union in deep neural networks for image segmentation". In: *International symposium on visual computing*. Springer. 2016, pp. 234–244.

[17] Jun Pan. "Recent Deep Neural Networks for Object Detection". In: *Highlights in Science, Engineering and Technology IFMPT 2022*. Vol. 31. Institute of International Education, School of Hebei University of Technology. Tianjin, China, 2023, pp. 268–270.

[18] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *Facebook AI Research (FAIR)* (2018). https://github.com/facebookresearch/Detectron. arXiv: 1708.02002v2 [cs.CV].

[19] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv preprint arXiv:1804.02767* (Apr. 2018). URL: https://pjreddie.com/yolo/.

[20] Dillon Reis et al. "Real-Time Flying Object Detection with YOLOv8". In: *Georgia Institute of Technology* (May 2023). URL: https://arxiv.org/abs/2305.09972v1.

[21] Ayush Yajnik. *Computer Vision: YOLO, Grid Cells, and Anchor Boxes*. Accessed: 2024-05-27. 2023. URL: https://medium.com/@ayushyajnik2/computer-vision-yolo-grid-cells-and-anchor-boxes-57b8a33cb25b.

[22] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. Accessed: 2024-05-27. 2016. URL: https://arxiv.org/abs/1506.02640.

[23] Minh-Tan Pham et al. "YOLO-Fine: One-Stage Detector of Small Objects Under Various Backgrounds in Remote Sensing Images". In: *Remote Sensing* 12.15 (2020). Accessed: 2024-05-27, p. 2501. URL: https://doi.org/10.3390/rs12152501.

[24] Panpan Gao et al. "Visual Question Answering for Intelligent Interaction". In: *Mobile Information Systems* 2022 (2022). DOI: 10.1155/2022/4232968. URL: https://doi.org/10.1155/2022/4232968.

[25] Ayisha Tabassum and Rajendra R Patil. "A survey on text pre-processing & feature extraction techniques in natural language processing". In: *International Research Journal of Engineering and Technology (IRJET)* 7.06 (2020), pp. 4864–4867.

[26] Kushal Kafle and Christopher Kanan. "Visual question answering: Datasets, algorithms, and future challenges". In: *Computer Vision and Image Understanding* 163 (2017). Language in Vision, pp. 3–20. ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2017.06.005. URL: https://www.sciencedirect.com/science/article/pii/S1077314217301170.

[27] Sara Silva, Rúben Pereira, and Ricardo Ribeiro. "Machine learning in incident categorization automation". In: *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE. 2018, pp. 1–6.

[28] Jian Xu et al. "Trouble ticket routing models and their applications". In: *IEEE Transactions on Network and Service Management* 15.2 (2018), pp. 530–543.

[29] Liang Peng et al. "Answer Again: Improving VQA With Cascaded-Answering Model". In: *IEEE Transactions on Knowledge and Data Engineering* 34.4 (2022), pp. 1644–1655. DOI: 10.1109/TKDE.2020.2998805.

[30] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[31] Jieh-Sheng Lee and Jieh Hsiang. "Patent claim generation by fine-tuning OpenAI GPT-2". In: *World Patent Information* 62 (2020), p. 101983. ISSN: 0172-2190. DOI: https://doi.org/10.1016/j.wpi.2020.101983. URL: https://www.sciencedirect.com/science/article/pii/S0172219019300766.

[32] Mohamed El Ghaly Beheitt and Moez Ben Hajhmida. "Automatic Arabic Poem Generation with GPT-2". In: Jan. 2022, pp. 366–374. DOI: 10.5220/0010847100003116.

[33] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2019). URL: https://arxiv.org/abs/1810.04805.

[34] Maryam Rahnemoonfar et al. "Floodnet: A high resolution aerial imagery dataset for post flood scene understanding". In: *IEEE Access* 9 (2021), pp. 89644–89654.

[35] Ajay Rawat. "A Review on Python Programming". In: *International Journal of Research in Engineering, Science and Management* 3.12 (2020), pp. 8–11. URL: https://www.ijresm.com/Vol.3_Issue.12_December2020/IJRESM_V3_I12_3.pdf.

[36] Tiago Carneiro et al. “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications”. In: *IEEE Access* 6 (2018), pp. 61677–61685. DOI: 10.1109/ACCESS.2018.2874767.

[37] Nikhil Ketkar and Jojo Moolayil. “Introduction to PyTorch”. In: *Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch*. Berkeley, CA: Apress, 2021, pp. 27–91. ISBN: 978-1-4842-5364-9. DOI: 10.1007/978-1-4842-5364-9_2. URL: https://doi.org/10.1007/978-1-4842-5364-9_2.