الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي

Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

N°Ref :……

Centre Universitaire Abdelhafid BOUSSOUF- Mila

Institut des Sciences et de la Technologie

Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de

Master

En : Informatique
Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)
Thème :

Drones Use in Disaster Management

Préparé par :

- Bennacer Nihad
- Amimour Nadjat

Soutenue devant le jury

Encadré par	**Dr. Aissa Boulmerka**	**Grade : Maître de conférences A**
Président	**Dr. Souheila Khalfi**	**Grade : Maître de conférences B**
Examinateur	**Mr. Dib Abderrahim**	**Grade : Maître assistant A**

Année Universitaire : 2022/2023

Abstract

This thesis explores the effective assessment of damages following natural disasters, such as floods, through the utilization of drone imagery. Deep learning techniques, particularly Artificial Neural Networks, have been extensively explored for tasks such as semantic image segmentation and object detection. The primary objective of this research is to identify and classify various types of damages resulting from disasters. The thesis evaluates several architectures of Artificial Neural Networks, including Unet + Resnet34 and Unet + VGG, in addition to implementing Deeplabv3 using the Floodnet dataset in the context of semantic image segmentation. As for the detection and classification of different types of damages, object detection techniques like YOLOv8n are applied using the Rescuenet dataset.

key words: deep learning, Convolutional neural networks, Semantic segmentation, Object detection, Unet, Deeplab, YOLOv8n.

ملخص

تستكشف هذه الأطروحة التقييم الفعال للأضرار الناجمة عن الكوارث الطبيعية، مثل الفيضانات، من خلال استخدام تحليل الصور الجوية. تم استكشاف تقنيات التعلم العميق، وخاصة الشبكات العصبية الاصطناعية، على نطاق واسع لمهام مثل تجزئة الصورة الدلالية واكتشاف الكائن. والهدف الرئيسي من هذا البحث هو تحديد وتصنيف مختلف أنواع الأضرار الناجمة عن الكوارث. تقيم الأطروحة العديد من بنى الشبكات العصبية الاصطناعية، بما في ذلك Unet + Resnet34 و Unet + VGG، بالإضافة إلى تطبيق Deeplabv3 باستخدام مجموعة البيانات Floodnet في سياق تجزئة الصور الدلالية. وفيما يتعلق باكتشاف وتصنيف أنواع مختلفة من الأضرار، يتم تطبيق تقنيات كشف الكائنات مثل YOLOv8n باستخدام مجموعة البيانات Rescuenet.

الكلمات الرئيسية - التعلم العميق، التجزئة الدلالية، الشبكات العصبية التلافيفية، اكتشاف الكائنات، Unet، Deeplabv، YOLOv8n.

Résumé

Cette thèse explore l'évaluation efficace des dommages après des catastrophes naturelles, telles que les inondations, grâce à l'utilisation d'images de drones. Les techniques d'apprentissage profond, en particulier les réseaux neuronaux artificiels, ont été largement explorées pour des tâches telles que la segmentation sémantique d'images et la détection d'objets. L'objectif principal de cette recherche est d'identifier et de classifier divers types de dommages résultant de catastrophes. La thèse évalue plusieurs architectures de réseaux neuronaux artificiels, notamment Unet + Resnet34 et Unet + VGG, ainsi que la mise en œuvre de Deeplabv3 en utilisant l'ensemble de données Floodnet dans le contexte de la segmentation sémantique d'images. 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 l'ensemble de données Rescuenet.

Mots-clés - apprentissage en profondeur, réseaux neuronaux convolutionnels, segmentation sémantique, détection d'objets, Unet, Deeplab, YOLOv8n.

Acknowledgement

First and foremost, I would like to express my gratitude to Allah for giving me the strength and determination to pursue my dreams despite facing numerous challenges in the past.

I am also thankful to Dr. Aissa Boulmerka for his valuable guidance, support, and expertise that enabled me to engage in a field that is of great interest to me. In addition, I am very grateful to my parents for their unwavering support, sacrifices, and prayers, as well as to my siblings and friends for being there for me throughout my journey.

I must also extend my sincere thanks to my colleague, Amimour Nadjat, who was my companion through all research struggles and sleepless nights. Finally, I am grateful to everyone who supported me, whether they were close relatives, teachers, or colleagues from the university staff.

Bennacer Nihad

I have always dreamed of conducting this humble work. I dedicate this achievement to my role model and support in life, my dear father, and to that great woman who raised and worked tirelessly, my loving mother. Also, to those who supported me and were a pillar for me, especially my husband and all members of my family, and especially my brothers who stood by my side throughout my journey.

I am grateful to Dr. Aissa Boulmerka for his valuable guidance, support, and expertise that allowed me to pursue a field that I am passionate about.

I would like to express my heartfelt gratitude to my colleague, Bennacer Nihad, who went through the same research challenges and long nights with me. Lastly, I am appreciative of all those who supported me, whether they were my close family members, mentors, or colleagues from the university.

Amimour Nadjat

Contents

List of Figures

List of Tables

List of Abbreviations

Adaboost	Adaptive Boosting
AI	Artificial intelligence
ASPP	Atrous Spatial Pyramid Pooling
ASIC	Application-Specific Integrated Circuits
Bn	Batch normalization
CNN	Convolutional Neural Network
CPU	Central Processing Units
DNN	Deep Neural Network
DL	Deep Learning
DPM	Deformable Parts Model
EMA	Exponential Moving Average
FC	fully connected
FCN	fully convolutional neural networks
FL	Focal Loss
IoU	Intersection-Over-Union
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
GPU	Graphics Processing Units
LiDAR	Light Detection and Ranging
LR	Learning rate
MLP	multi-layer perceptrons
ReLU	Rectified Linear Unit
R-CNN	Region-based Convolutional Neural Networks
RPAS	Remotely Piloted Aircraft Systems
RPN	Region Proposal Network
RPV	Remotely Piloted Vehicles
SSD	Single Shot Detector
SVM	Support Vector Machine

Tanh	The Hyperbolic Tan
TPU	Tensor Processing Units
UAS	Unmanned Aerial Systems
UAV	Unmanned Aerial Vehicle
VGG	Visual Geometry Group
YOLO	You Only Look Once

General introduction

Disasters pose significant challenges in terms of damage assessment and timely response. Using Unmanned Aerial Vehicles (UAV), or drones, has emerged as a rapid and effective solution to support disaster management efforts. These aerial platforms can assist in capturing high-resolution images and gathering critical data from inaccessible or hazardous areas.

Deep learning techniques, including object detection and semantic segmentation, play a vital role in enhancing drones' disaster management capabilities. Object detection algorithms enable drones to identify and locate specific objects or structures of interest, such as damaged buildings or infrastructure. This information helps in prioritizing response efforts and allocating resources efficiently.

This thesis explores essential concepts and methodologies for implementing image semantic segmentation and object detection. The utilization of these models significantly enhances the accuracy and efficiency of disaster management operations. UNet, DeepLab, and YOLOv8n have demonstrated remarkable capabilities in accurately identifying and classifying objects in drone imagery, which is crucial for effective disaster response and recovery efforts. The following is a summary of the main topics covered in this master's thesis:

In the first chapter, we explore the definition of a disaster and provide examples of various types, including earthquakes, floods, and forest fires. Additionally, we delve into the concept of Disaster Management, defining its purpose and significance. We also examine the definition of drones and discuss the reasons for their utilization in Disaster Management. Furthermore, we present examples of Disaster Management Drone applications and discuss the components of a drone. The challenges faced by UAVs are also addressed. Moreover, we compare the capabilities of satellites, aircraft, and UAVs. Lastly, we explore the analysis of drone-captured images for surveillance purposes, including techniques such as semantic segmentation, image classification, object detection, and Visual Question Answering.

We begin the second chapter by presenting an overview of DL concepts, starting with fundamental principles of artificial neural networks, such as perceptrons, multilayer perceptrons, activation functions, cost functions, gradient descent, learning rate, and the important processes of forward propagation and backpropagation. Additionally, the chapter explores widely used deep neural network architectures in computer vision, specifically focusing on CNNs. It discusses notable CNN architectures, including LeNet5, AlexNet, VGG16, GoogLeNet, and ResNet.

In the third chapter, our focus will be on two significant tasks within computer vision: semantic segmentation and object detection. 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 and its variants, DeepLab, YOLO, RCNN, and SSD. By examining these architectures, we will uncover their respective strengths and weaknesses.

In the final chapter, we focus on the application of semantic segmentation and object detection techniques to drone images. We start by defining the problem we aim to solve in this context. Next, we provide a detailed description of the dataset we will use and how it will be augmented. We next go to the network design details employed to solve the problem and how they were trained. After that, we conclude by describing our experiments, tests, and results. The obtained results are reported in tables and evolution curves of the metrics of each model. Finally, we will describe some of the most popular deep learning hardware and frameworks/tools, including the Python programming language, Google Colaboratory, TensorFlow, Keras, OpenCV, Roboflow, and ultralytics.

Chapter 1

Drones Use in Disaster Management

Introduction

A disaster is a major disturbance of a community or society's ability to operate that causes broad consequences on people, property, the economy, or the environment and which surpasses the capacity of the afflicted community or society to deal with using its resources [1]. Tracking disasters is important for mitigating their impact and damage on the environment and population. This can be made easier by using Unmanned Aerial Vehicles (UAVs) with cameras to take images of interesting places. The proliferation of UAVs in several fields has stimulated research into their potential benefits in disaster management. Drones create a communication network between victims, survivors, and designated rescue teams. This chapter will cover disasters in general, give some examples, and go into deeper detail about drones.

1.1 Definition of disasters

Disaster can be defined as a natural or artificial event that may cause it or result from a deliberate act by man. These occurrences can include forest fires, earthquakes, flooding, and explosions. Disasters can also have long-term impacts on communities and the environment, leading to loss of life, displacement, economic and social disruption, and psychological trauma [2].

People, communities, and governments must be proactive to prepare for disasters. This includes developing emergency plans and stockpiling resources like food, water, and medical equipment. Governments can also spend money on early warning systems and disaster-proof infrastructure to mitigate the effects of disasters. Also, it is essential to move quickly to restore critical infrastructure, such as water and electrical services, to support communities in recuperating and returning to normal. To mitigate their impacts and help affected people, coping with catastrophes is generally challenging and requires cooperation and coordination between individuals, groups, and governmental entities.

1.2 Some examples of disasters

1.2.1 Earthquakes

An earthquake is a typical disaster that breaks out suddenly without any warning, causing serious damage to buildings and killing many people. The chances of survival for someone trapped in a collapsed building depend largely on the damage to the affected building. Therefore, it is essential to quickly map the affected areas to assess the damage and, more importantly, optimize the sharing of rescue resources [3].

Rescue operations should be prioritized according to the extent of the damage and the likelihood of finding survivors. In some circumstances, it might be necessary to concentrate on saving those who are seriously hurt or in danger, whereas, in others, it might be more beneficial to concentrate on removing debris to open up access to areas where survivors are more likely to be found.

Figure 1.1: An image of an earthquake that occurred in Turkey.

1.2.2 Forest fires

In forests, wildfires represent a prevalent danger. There exist three stages of fire management: before the fire, during the fire, and following the fire. Historical fire detection techniques relied on manual labor. Satellite-linked monitoring methods are the go-to tool for spotting forest fires. However, their applicability is limited to substantial areas. Detection of fires frequently occurs too late, resulting in forest loss. As a consequence, current methods could be more successful. Research shows that nearly 80 percent of forest damage can be attributed to this issue [4].

Various studies have suggested solutions for monitoring forests after a fire, utilizing different tools such as drone images. UAVs have numerous advantages, such as their efficiency and ability to produce high-resolution images with great temporal and spatial accuracy, making them an effective remote sensing system. With the support of ground stations and technological advancements utilizing big data and

artificial intelligence, Post-processing of UAV-collected data is possible in almost real-time, providing significant benefits as a monitoring system [5].

Figure 1.2: Close-up images of forest fires captured by drones.

1.2.3 Floods

Flooding is a common natural disaster that disproportionately impacts low-income nations [6]. This is caused by several factors, including (i) the lack of infrastructure and resources for preparing for and responding to flooding, (ii) higher population densities, (iii) and a greater reliance on farming and other activities that are susceptible to flooding. A cycle of poverty and ongoing vulnerability results when low-income countries frequently lack the financial resources to repair the harm caused by floods. Flooding's effects on low-income countries must be addressed holistically, with investments made in infrastructure, early warning systems, and the development of resilient communities.

Drones can be a very beneficial tool for managing floods. With their ability to capture high-resolution images and videos, drones can deliver in-the-moment information in depth about the scope and severity of flooding. This can be invaluable for emergency response teams, who need to quickly assess the situation and make informed decisions about where to allocate resources. They are a valuable tool in flood management, providing critical information that can help emergency respondents to make informed decisions and optimize their resources [7].

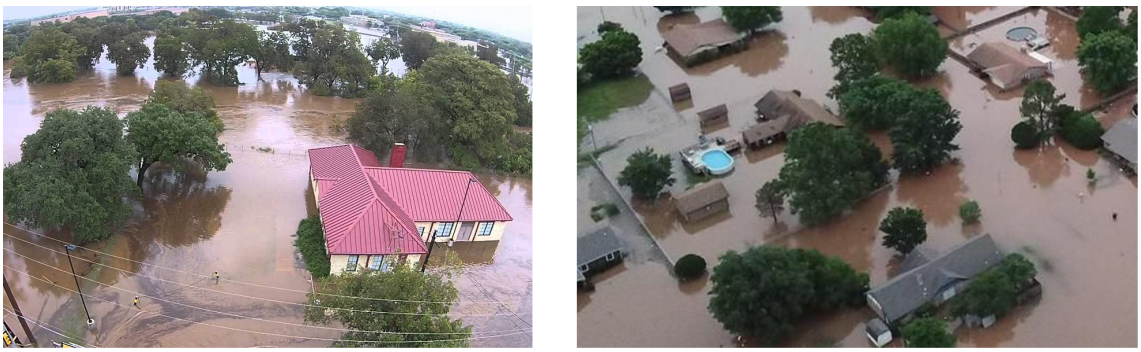


Figure 1.3: Close-up images of floods captured by drones.

1.3 Disaster management

Disaster management is a planned process of preparing for, dealing with, and recovering from natural and man-made disasters [8].

Disaster management aims to ensure that affected communities can promptly recover and rebuild after a disaster and lessen the harmful effects of disasters on people and the environment. Based on the risk assessment, emergency planners develop disaster preparedness plans and strategies, including training and drills, emergency response protocols, and communication systems, to ensure that information can be quickly shared with relevant stakeholders.

Regarding rapid disaster response and recovery, drones are considered one of the most effective solutions.

UAVs can make surveillance and monitoring tasks easier by performing tasks like object detection, activity recognition, and search and rescue operations. With the help of advanced machine learning and artificial intelligence algorithms, drones can access previously inaccessible areas and provide more comprehensive coverage. Compared to traditional aerial data collection methods like helicopter surveys and flyovers, using UAVs for data collection is more cost-effective and requires fewer resources. Moreover, drones can capture high-resolution video sequences and still images from locations that are challenging to reach for helicopters and humans [9].

1.4 Definition of a drone

Drones, which are also known as Unmanned Aerial Vehicles (UAVs), can be referred to by a variety of names, including Unmanned Aerial Systems (UAS), Remotely Piloted Vehicles (RPV), and Remotely Piloted Aircraft Systems (RPAS). While "drone" is more widely recognized, other terms such as UAVs, RPA, RPAS, and UAS are official names that describe this technology depending on the jurisdiction [10].

UAVs equipped with camera sensors can provide valuable insights and observations in remote or inaccessible areas, making them useful tools for emergency and disaster management applications [11].

In the case of natural disasters such as floods, forest fires, traffic incidents, or earthquakes, By utilizing UAVs, it is possible to conduct rapid assessments of the damage caused by a disaster and promptly identify areas that require urgent attention. They can also be used to search for survivors or assess the safety of structures that may be unstable or at risk of collapse.

Drones have become an effective way to provide high-resolution images, especially for capturing images in areas that are difficult to access or where satellite and closed-circuit television cameras are ineffective [11].

Figure 1.4: Drone over a disaster area.

1.5 Reasons for using drones in disaster management

- Drones have become essential for disaster management and search and rescue operations.
- Drones can be deployed quickly in areas that are too dangerous for humans, providing valuable assistance and information to first responders.
- Drones equipped with augmented reality technology can provide real-time information and guidance to rescuers, allowing them to navigate through hazardous terrain or identify potential dangers.
- Drones with sensors such as infrared cameras can detect heat signatures, allowing rescuers to locate individuals who may be trapped or lost.
- In addition to their agility and accessibility, drones are much more cost-effective than traditional methods such as helicopters.
- Drones can cover large areas quickly and efficiently, providing valuable information to emergency responders without breaking the bank.

Drones have proven valuable tools for disaster management, search and rescue operations, and many other applications. As technology advances, we expect to see more capabilities and features added to these versatile machines [12].

1.6 Examples of disaster management drone applications

These examples demonstrate some of the many ways in which drones can be utilized in disaster management:

1.6.1 Damage assessment

Drones can be utilized for rapid surveys and to assess the extent of damage caused by a disaster, such as a flood, earthquake, or hurricane. This helps emergency responders to plan their response and allocate resources efficiently [13].

1.6.2 Search and rescue

Drones can be employed to search for survivors in areas that are hazardous or inaccessible for humans, such as collapsed buildings, flooded areas, or forested regions [14]. Drones equipped with high-resolution cameras and thermal imaging sensors can search for survivors in areas that may be too dangerous for humans to access. They can quickly cover large areas and provide real-time footage to aid search and rescue efforts. Additionally, drones can be equipped with speakers or microphones to communicate with survivors or to broadcast messages to a larger area. This can help locate and rescue survivors more quickly, ultimately increasing the chances of survival.

1.6.3 Mapping and monitoring

. Drones can be equipped with video cameras, thermal cameras, laser scanners, and other sensors to capture high-resolution images and data from disaster-affected areas and carry out the search and rescue mission [15]. This information can be used to create 3D models, topographical maps, and other visualizations that help emergency responders plan their operations and prioritize their efforts. On the other hand, they are typically used to observe ongoing disasters and provide real-time updates to response teams. For example, drones can track the spread of a forest fire and identify areas where the fire is particularly intense. This information can guide firefighting efforts and help protect nearby communities.

1.6.4 Communication

Drones could be key in building communication networks in disaster areas. Traditional communications infrastructure is often damaged or destroyed in the aftermath of disasters, leaving responders and survivors without reliable communication. This is where drones equipped with communications equipment can help. They can quickly and easily establish temporary communications networks in disaster areas, allowing rescue and recovery teams to coordinate their efforts and help those in need. In other words, drones can help bridge the communications gap in disaster-stricken areas where traditional networks have failed. Therefore, a reliable emergency communication system after a disaster is essential, and drones can be an invaluable tool in building and maintaining such a system [16].

1.6.5 Delivery of aid and supplies

Drones have the potential to transport essential aid and provisions, such as nourishment, clean water, and medical resources, to regions impacted by disasters that are challenging to reach through alternative methods.

1.6.6 Risk assessment

UAVs can assist in evaluating the extent of damage caused by disasters using various techniques, including structural health monitoring and video inspection conducted by UAVs [17].

1.7 Component of drone

A drone that is designed for disaster management purposes is usually comprised of several distinct components.

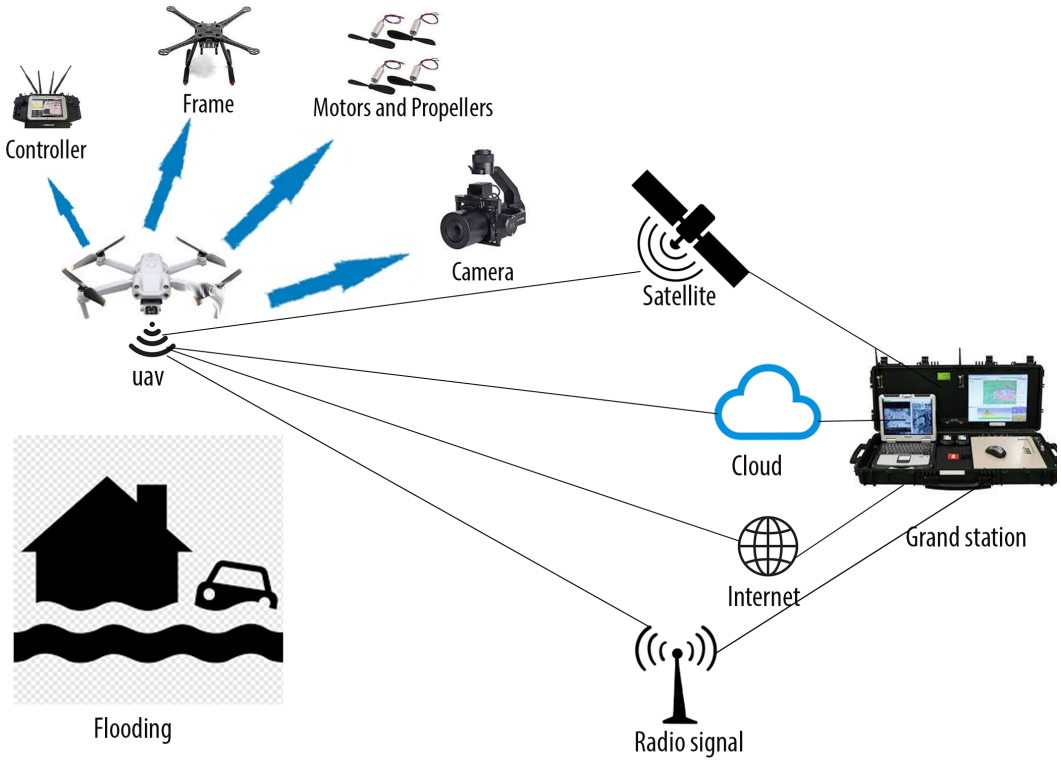


Figure 1.5: The essential components of drones for effective emergency management.

These components are essential for the drone to function effectively and efficiently in emergencies:

1.7.1 Frame

The frame provides the structure and support for the drone's other components. It must be lightweight, durable, and able to accommodate the payload. The frame of a drone for disaster management can be made of different materials, including aluminum or carbon fiber. Aluminum frames are commonly used because they are lightweight and durable, making them suitable for rugged environments. On the other hand, carbon fiber frames are even lighter than aluminum and provide greater rigidity and strength, but they can be more expensive. The choice of material will depend on the specific requirements of the drone and the budget available for the

project [18].

The classification of drone frame construction is often based on the number of arms, as the number of arms typically determines the drone's stability, maneuverability, and payload capacity.

1.7.2 Motors and propellers

The motors and propellers provide the necessary lift and propulsion for the drone to fly. They must be powerful and reliable to withstand adverse weather conditions and carry the payload.

1.7.3 UAV controller

Effective control technology design and analysis are crucial for managing drone disasters, as drones are inherently unstable and pose significant risks when not adequately controlled. To reduce the potential hazards related to drones, it is essential to comprehend stability concerns and apply a rigorous approach to control technology design. A comprehensive treatment of UAV control and related technologies is essential to address the challenges of controlling drones in emergencies and avoiding disasters [19].

1.7.4 Sensors

Sensors and sensing strategies are essential for an unmanned aircraft to perceive and comprehend the environment around it, allowing it to function intelligently without a human pilot onboard. These technologies enable drones to "sense," "see," "hear," and "understand" their surroundings in unknown and cluttered environments. In essence, sensors and sensing strategies are crucial for unmanned aircraft operating as if a human pilot were onboard [19].

1.7.5 Camera and other payloads

Cameras and other payloads, such as thermal cameras, multispectral cameras, gas sensors, and lidar, are crucial in providing situational awareness and gathering important data during disaster management operations.

1.7.6 Battery

The battery is the power source for the drone and provides the energy needed to operate in the air. Longer battery life is especially important for drones used in disaster management operations as they need to remain in the air for extended periods.

1.7.7 Communication system

The communication system is a crucial link between the drone and the ground station, enabling the operator to control the drone and receive real-time data. Various communication systems are commonly used in the aviation industry for air-to-

ground communication, including Voice Communication, Satellite Communication, the internet, and the cloud.

1.8 Aerial photography factors

When it comes to aerial photography with a drone, several factors can affect the quality and usefulness of the captured images. Here are some important ones:

1.8.1 Image resolution

A higher resolution image means more detail and clarity, which can be important for certain applications such as surveying or inspection. Choosing a drone with a camera with the appropriate resolution for the intended use is important.

1.8.2 Image size and aspect ratio

Depending on the intended use of the images, it may be necessary to consider the size and aspect ratio of the images. For example, if the images will be used for marketing or advertising purposes, they may need to conform to certain size or aspect ratio standards.

1.8.3 Image format

The format of the image can affect its usability and compatibility with different software programs. Common formats for aerial images include JPEG, TIFF, and RAW.

1.8.4 Image stabilization

A drone with image stabilization technology can help reduce the impact of wind or other disturbances on the image quality, resulting in sharper and clearer images.

1.8.5 Field of view and lens quality

The field of view of the camera lens can impact how much of the scene is captured in the image. Additionally, the lens's quality can affect the image's sharpness and clarity.

1.9 UAV challenges

Despite UAVs occupying a significant portion of the market by 2025, numerous obstacles remain to overcome in their development and control. This section will explore the various research challenges that scientists and renowned researchers face [20].

1.9.1 Security concerns

Security concerns in disaster management with UAVs are also significant. One of the primary concerns is protecting sensitive data collected by UAVs during disaster response operations, such as personal information or images of damaged infrastructure. This data can be used maliciously if it falls into the wrong hands for identity theft or to gain access to critical infrastructure.

Another security concern is the potential use of UAVs to disrupt disaster response operations. For example, someone could fly a drone in the area where first responders are working, potentially causing accidents or interfering with the operation of other UAVs or equipment.

In addition to security concerns related to data and operations, cyber attacks on UAV control systems are also risky. A cyber attack on a UAV's control system could cause the vehicle to malfunction or crash, potentially causing harm to people or property [21].

To address these security concerns, developing and implementing appropriate security protocols and guidelines for using UAVs in disaster response operations is essential. This may include measures such as encryption of data, securing control systems, and implementing no-fly zones around disaster response operations.

1.9.2 Safety concerns

The incidents involving UAVs have highlighted several safety concerns, which can be broadly categorized into four areas. These include UAV design issues, operational standards, technological requirements for their safe operation, the risk of signal interference or hijacking, and the necessity of government regulations and public awareness to ensure their secure usage [20].

1.9.3 Privacy concerns

UAVs can collect vast amounts of data through images, videos, and mapping coordinates, typically transmitted to users through a network. However, the transmission and storage of this data can give rise to privacy concerns and issues. For instance, there is a risk of location privacy breaches, linking attacks, man-in-the-middle attacks, eavesdropping attacks, and other privacy violations [20].

1.9.4 Scalability issues

Scalability is a critical issue when using UAVs in a large-scale operation or deployment. As the quantity of UAVs grows, managing and controlling them effectively becomes challenging. This problem is known as the scalability issue and is a significant challenge for researchers and scientists.

One of the primary scalability issues is the ability to maintain control and communication with large numbers of UAVs simultaneously. As the quantity of UAVs grows, it is essential for communication and control systems to keep up with the

increased traffic and maintain dependable connections.

As UAV swarms continue to grow, even new developments like the blockchain platform may not be suitable for achieving scalable communication. While some platforms have been developed that use consensus algorithms to improve transmission speed, they may not be suitable for UAV swarms, which are becoming increasingly common in smart cities. Therefore, there is a need to develop new communication platforms that can support the growing number of UAVs in swarms and ensure reliable, fast communication between them [20].

1.9.5 Limited Resources

The limited resources problem in drones can cause various challenges and limitations. For example, drones with limited energy reserves may have a shorter operational time, which reduces their effectiveness in completing tasks that require extended flights.

In addition limited storage capacity may restrict the amount of data that can be captured, analyzed, and transmitted, affecting the drone's ability to provide accurate and timely information. Limited computational power can also be an issue when performing complex data analysis tasks, which may require higher processing speeds and more energy [20]. These limitations can hinder the ability of drones to perform critical tasks in industries such as disaster management, where real-time data and support are crucial.

1.10 Comparison of Satellite, Aircraft, and UAV

1.10.1 Satellite

Satellites capture images of the Earth's surface using sensors and cameras mounted on board. These sensors are sensitive to different wavelengths of light, allowing satellites to capture images in different spectral bands [22], such as visible, infrared, and microwave. Satellites orbit the Earth at fixed positions, and as the Earth rotates, the satellite captures images of different areas. Satellites can capture large areas of the Earth's surface in a single pass, and the images are typical of high spatial resolution, making them useful for various applications, such as land cover mapping, disaster monitoring, and environmental monitoring.

1.10.2 Aircraft

Aircraft, such as airplanes and helicopters, are equipped with sensors and cameras to take pictures of the surface of the Earth. Aircraft fly at different altitudes and speeds depending on the type of sensor being used and the application. For example, if high spatial resolution imagery is needed, the aircraft will fly at a lower altitude and slower speed. Aircraft are useful for capturing smaller areas compared to satellites, but they are more flexible in flying to specific locations and capturing imagery on demand.

1.10.3 Unmanned Aerial Vehicle

UAVs are small aircraft utilized remotely or autonomously without a human pilot on board. UAVs have cameras and sensors to capture high-resolution imagery of the Earth's surface. Like aircraft, These sensors can include digital cameras, multispectral cameras, and LiDAR (Light Detection and Ranging) sensors, among others [23].

UAVs can fly at lower altitudes and slower speeds than satellites, providing a more detailed and closer view of the Earth's surface. UAVs are also more flexible than satellites, as they can be launched and flown on demand, making them suitable for applications that require timely responses, like disaster management.

1.11 Analyzing drone-captured images for surveillance

One issue with using image drones is the sheer volume of data they can capture quickly. This can pose a challenge for humans tasked with analyzing and extracting pertinent information from the data, particularly in time-sensitive situations such as disaster response. The timely and accurate data analysis is crucial in such applications, making it imperative to develop efficient and effective methods for processing large amounts of data captured by image drones.

Here are some ways that segmentation, classification, Visual Question Answering, and object detection can help address these challenges:

1.11.1 Semantics segmentation

Image segmentation, particularly semantic segmentation, is vital in computer vision. It involves dividing an image into distinct segments or regions based on color, texture, or shape criteria. In semantic segmentation, each pixel in the image is assigned a specific label or class, providing a detailed understanding of the image's content. This process allows for precise object recognition, scene understanding, and accurate delineation of different regions within the image. Semantic segmentation finds applications in numerous fields, including medical image analysis, autonomous vehicles, video surveillance, and augmented reality [24].

1.11.2 Image Classification

A classification method is employed to classify a given data point into a specific category or group. It involves combining relevant features to represent the object and comparing it with a pre-trained model. In visual recognition, classification is essential for distinguishing the target object from other objects, resulting in a more meaningful and descriptive representation of the object. Popular classification methods include SVM, DPM, and AdaBoost [25].

Figure 1.6: Sementic segmentation of flooded aerial,(a)Image Original,(b)Image Segmentation

.

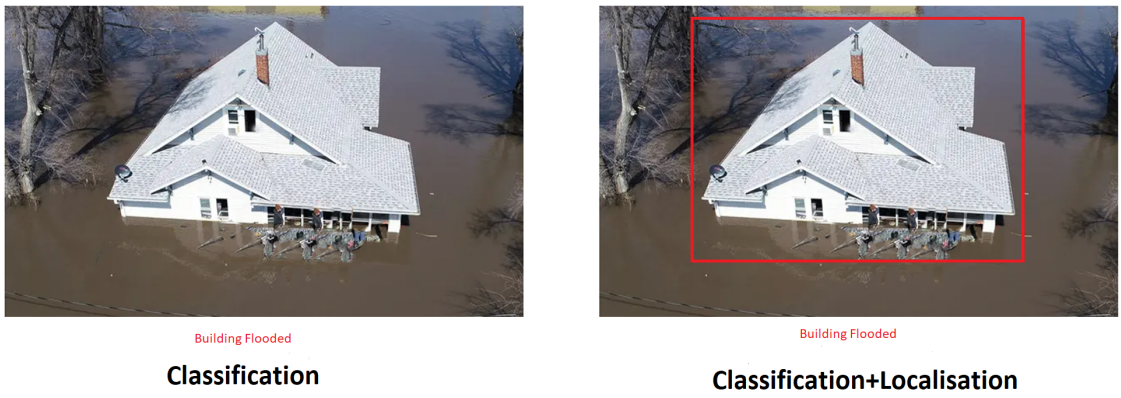


Figure 1.7: Image Classification

1.11.3 Object Detection

Object detection is the process of identifying and localizing objects within an image or video. It involves detecting the presence of objects in an image or video and accurately localizing them by drawing bounding boxes around them. Object detection aims to automatically and accurately identify the location and type of objects present in an image or video, which is a crucial task in many applications, such as autonomous driving, surveillance, and robotics.

The evolution of object detection techniques, from traditional methods based on handcrafted features to the more recent deep learning-based approaches, has significantly improved the accuracy and efficiency of object detection [26].

1.11.4 Visual Question Answering

Visual Question Answering (VQA) is a field of artificial intelligence that involves training a model to answer questions about visual content, such as images or videos, using textual responses. It combines computer vision and natural language processing techniques to enable machines to understand and generate answers to questions about visual content.

Figure 1.8: An Example of Object Detection

Conclusion

This chapter focused on the potential of drones for disaster management, covering their ability to assist in damage assessment, search and rescue operations, and supply delivery. The following chapter will explore the most common deep learning concepts and their various applications in this field.

Chapter 2

Deep Learning

Introduction

The foundational principles of deep learning (DL) have been shaped by the structure of the human brain, aiming to replicate human-like learning. Consequently, several fundamental terms in DL can be traced back to neurology. This chapter introduces the concepts of DL. The chapter discusses general concepts of artificial neural networks, such as perceptrons, multilayer perceptrons, activation functions, cost functions, gradient descent, learning rate, forward propagation, and backpropagation. Furthermore, the chapter explores the commonly used deep neural network architectures in the computer vision field, such as convolutional neural networks and Common convolutional neural network architectures.

2.1 Deep Learning

Deep neural networks originated from modeling biological vision and brain information processing [27]. DL is a branch of machine learning and artificial intelligence (2.2) [28]. DL is a subfield of machine learning that utilizes neural networks with multiple layers to learn and extract features from data. This approach has proven to be highly effective compared to traditional machine learning methods [29].

His technology has significantly improved the accuracy of various computer vision tasks, including image segmentation, object detection, and classification. In image segmentation, deep learning models use large amounts of annotated image data to learn how to segment images into smaller regions or segments for analysis. In object detection, deep learning models use CNNs to identify and locate objects within an image. Deep learning has also been successful in classification tasks, where it can accurately classify images or other data into predefined categories.

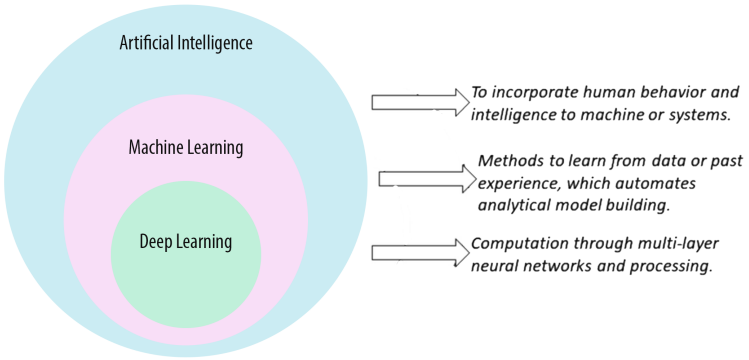


Figure 2.1: An illustration of the position of DL, ML and artificial intelligence (AI) [28].

2.2 Artificial neural networks

Neural networks are a type of computational model inspired by the structure and function of the biological brain. They consist of many interconnected processing units, called neurons, that work together to perform complex computations in parallel. The connections between neurons are modeled as numerical weights, determining the strength of the signal transmitted between them.

The architecture of a neural network is determined by several key factors, including the number of neurons, the number of layers, and the types of connections between layers [30].

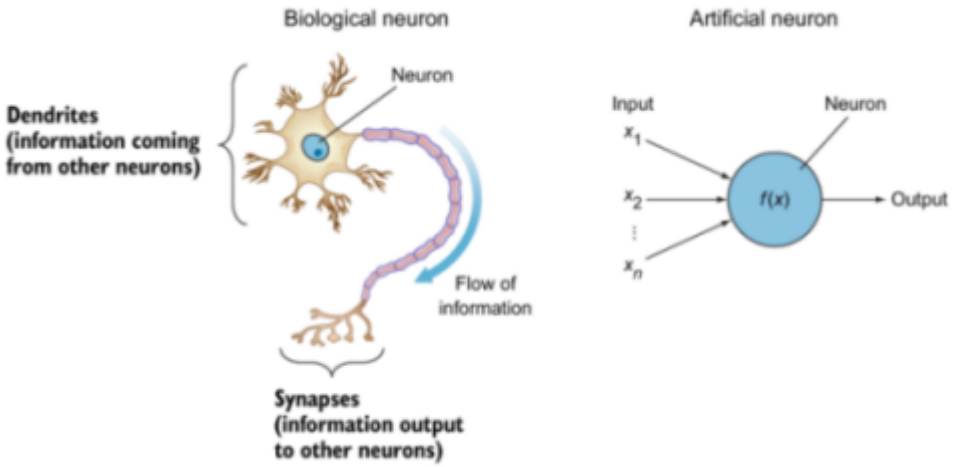


Figure 2.2: Artificial neurons were inspired by biological neurons. Different neurons are connected by synapses that carry information [31].

2.3 Perceptrons

The perceptron is the simplest type of neural network, consisting of a single artificial neuron. Its function is conceptually similar to that of a biological neuron, which

receives electrical signals from its dendrites, modulates the signals, and fires an output signal only when the total strength of the inputs exceeds a certain threshold.

The perceptron carries out two main operations to simulate this behavior in an artificial neuron. Firstly, it computes the weighted sum of the input values to quantify the overall strength of the input signals. Secondly, it applies a step function to the result to determine whether to generate an output signal. If the computed signal surpasses a specific threshold, the output signal will be 1; otherwise, if the signal does not surpass the threshold, the output signal will be 0 [31].

The perceptron is frequently employed for tasks involving binary classification, where it learns to distinguish between two classes of data by iteratively adjusting its weights to reduce the difference between its predicted outputs and the actual outputs. However, due to its limited capacity to capture intricate relationships between inputs, the perceptron is a basic building block for more sophisticated neural network architectures.

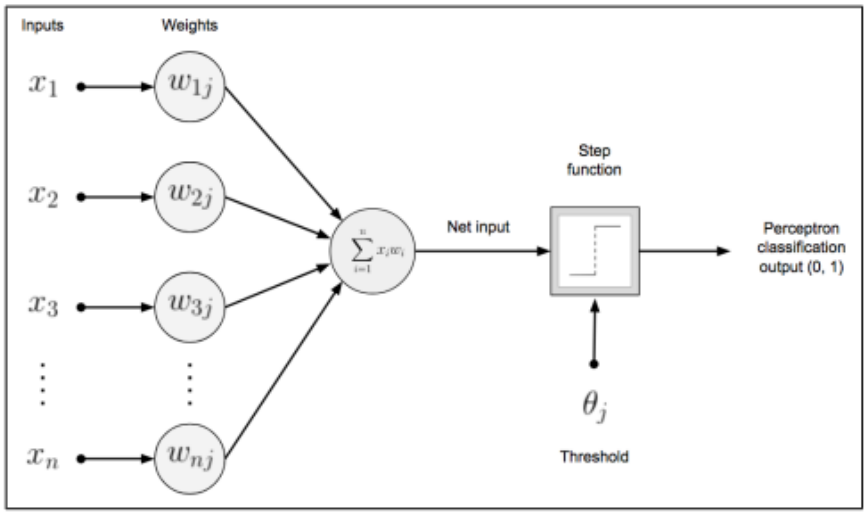


Figure 2.3: Single-layer perceptron [30].

2.4 Multilayer Feed-Forward Networks

The multilayer feedforward neural network, such as multilayer perceptrons (MLPs), consists of multiple layers of artificial neurons, including the input, hidden, and output layers, as seen in 2.4.

Each layer contains one or more artificial neurons that resemble the perceptron model but have a distinct activation function depending on the particular layer's function in the network [30].

The input layer receives data from an external source, and the hidden layers apply transformations to the input data using weighted connections between neurons. Each hidden neuron applies a unique activation function, such as ReLU or sigmoid,

to the weighted sum of its inputs. The output layer applies the final transformation to the output of the last hidden layer. The activation function used for this layer is chosen based on the specific task of the neural network. During training, the network's weights are adjusted using an optimization method, such as backpropagation, that minimizes a cost function.

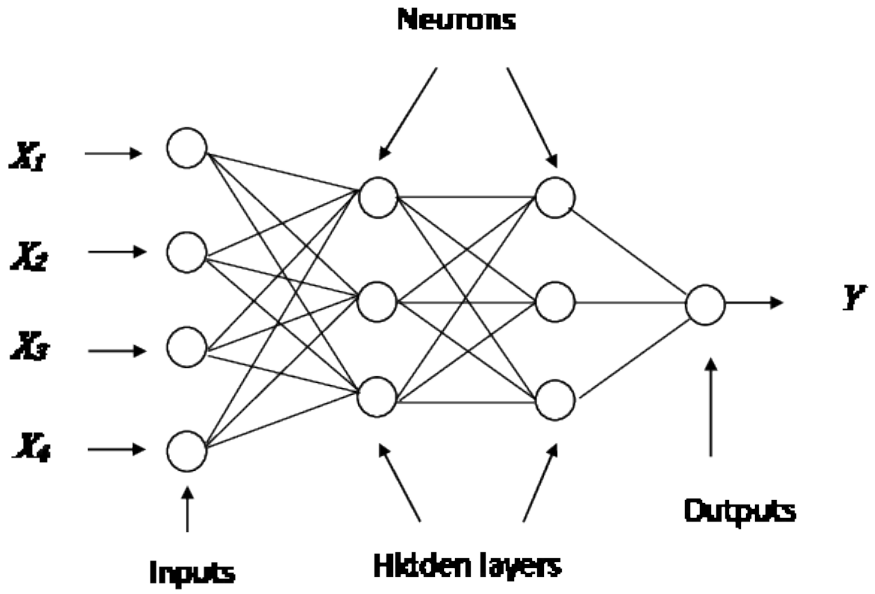


Figure 2.4: Topology of a multilayer neural network [30].

2.5 Activation functions

An activation function is a crucial element in a neural network, as it transforms an input signal into an output signal fed to the next layer in the network [32].

Without an activation function, a neural network would be equivalent to a linear regression model, which performs a linear transformation without the ability to capture nonlinear patterns. Some popular activation functions include the Sigmoid function, Tanh, ReLU, and Softmax.

2.5.1 Sigmoids

The sigmoid function is one of the most commonly used activation functions in neural networks [33]. It is defined as follows:

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

Ṫhe sigmoid function maps any real-valued number to a value between 0 and 1, resulting in a continuous range of output values.

2.5.2 Tanh

The Hyperbolic Tanh, also called a symmetric sigmoid, is steeper than that of the sigmoid function, which means that it can produce more pronounced output values. It is defined as follows:

$$tanh(x) = 2sigmoid(2x) - 1$$

Which is:

$$f(x) = tanh(x) = \frac{2}{1 + e^{-}2x} - 1$$

2.5.3 ReLU

ReLU is another popular function, and it is preferred over sigmoid in recent networks. It is defined as follows:

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

This means that if the input x is less than 0, the output is 0; if it is greater than 0, the output equals the input. The derivative of ReLU is 1 for x $>$ 0 and 0 for x $<=$ 0. ReLU avoids very small values and returns either 0 (which can cause some gradients to vanish) or 1 [34].

2.5.4 Softmax

The softmax function is commonly employed in multi-class classification tasks and extends the sigmoid function. It ensures that the outputs of a neural network sum up to 1 and produce probabilities for each class, where each probability value falls between 0 and 1. The softmax function is commonly used in deep learning problems where there are multiple classes to predict, and we need to choose the most probable class among them [31]. It is defined as follows:

$$S(x_i) = \frac{e_i^x}{\sum_{j=1}^{n} e_j^x}$$

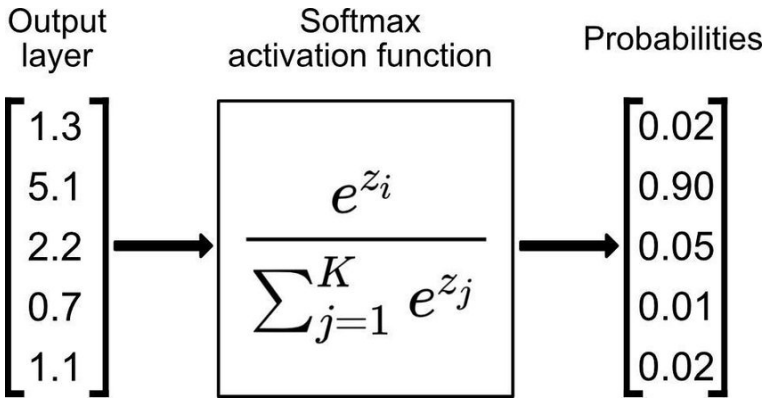


Figure 2.5: The softmax function transforms the input.

2.6 Batch normalization

Batch normalization (BN) is a popular technique used in deep learning to normalize the activations of the neurons in intermediate layers of a neural network [35].

Batch normalization is an algorithmic technique that enhances the stability and expedites the training process of DNN [36]. After BN alters the signal at each hidden layer, it looks like this:

$$(1) \quad \mu = \frac{1}{n}\sum_i Z^{(i)} \qquad (2) \quad \sigma = \frac{1}{n}\sum_i (Z^{(i)} - \mu)$$

$$(3) \quad Z^{(i)}_{norm} = \frac{(Z^{(i)} - \mu)}{\sqrt{\sigma^2 - \varepsilon}} \qquad (4) \quad \breve{Z} = \gamma * Z^{(i)}_{norm} + \beta$$

Using (1) and (2), the BN layer first calculates the mean μ and standard deviation σ of the activation values throughout the batch (2). The activation vector $Z^{(i)}_{norm}$ is then normalized with (3). As a result, the output of each neuron follows a conventional normal distribution across the batch (For numerical stability, ε is utilized as a constant).

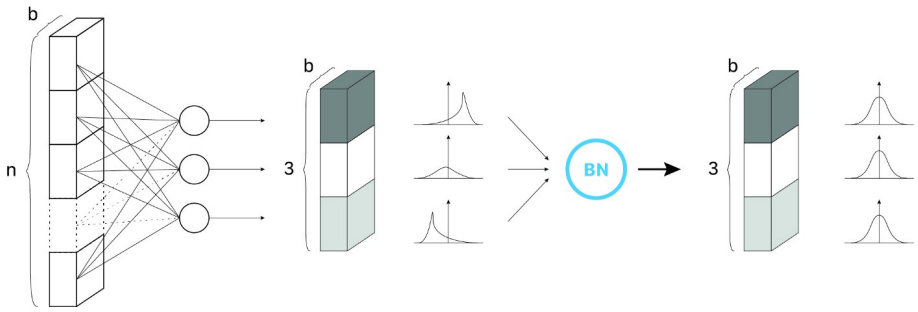


Figure 2.6: Batch normalization first step. Example of a 3-neuron hidden layer, with a batch of size b. Each neuron follows a standard normal distribution from [37].

By applying a linear transformation with two trainable parameters γ and β, it calculates the layer's output $Z^{(i)}_{norm}$ at the end (4). This phase allows the model to select the best distribution for each hidden layer by modifying two parameters: γ allows to alter the standard deviation; β permits to alter the bias by moving the curve to the right or left.

The network calculates the mean μ and standard deviation σ for the current batch at each iteration. When γ and β are ready, they are trained using gradient descent and an EMA to provide more weight to recent iterations.

2.6.1 Regression losses

- Mean Square Error/Quadratic Loss/L2 Loss

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2)}{n}$$

- Mean Absolute Error/L1 Loss

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n}$$

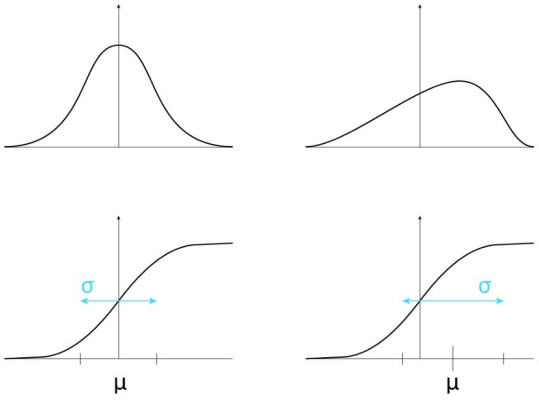


Figure 2.7: Benefits of γ and β parameters. Modifying the distribution (on the top) allows us to use different regimes of the nonlinear functions (on the bottom) from [37].

- Mean Bias Error

$$MBE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i))}{n}$$

2.6.2 Classification losses

- Hinge Loss/Multi class SVM Loss

$$SVMLOSS = \sum_{j \neq y_i} max(0, s_j - s_y i + 1)$$

- Cross Entropy Loss/Negative Log Likelihood

$$CrossEntropyLoss = -(y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i))$$

- Focal Loss

$$FocalLoss = FL(p_t) = -(1 - p_t)^{\gamma} log(p_t)$$

The FL loss function is an extension of the binary cross-entropy loss function used in binary classification. It introduces a focusing parameter that increases the penalty for misclassifying difficult samples compared to easier ones [38].

2.7 Loss functions

Loss functions are used to evaluate the performance of a neural network by measuring how close its predicted output is to the expected output. The goal is to minimize the difference between the predicted and expected outputs. This is achieved by finding the optimal values for the parameters, such as weights and biases, that minimize the loss function [30]. Different types of loss functions are used based on the type of problem, such as regression or classification.

2.8 Hyperparameters

Hyperparameters in deep learning models refer to the parameters not learned during training but set by the developer before the training process. These hyperparameters can significantly impact the model's performance, and finding the optimal values for them is crucial to achieving the best results.

2.8.1 Gradient descent algorithms

Gradient descent is a widely used optimization algorithm and a popular choice for optimizing neural networks [39]. Gradient descent has three variants that differ in how much data is used to compute the gradient of the objective function.

- Batch gradient descent

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

- Stochastic gradient descent

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

- Mini-batch gradient descent

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

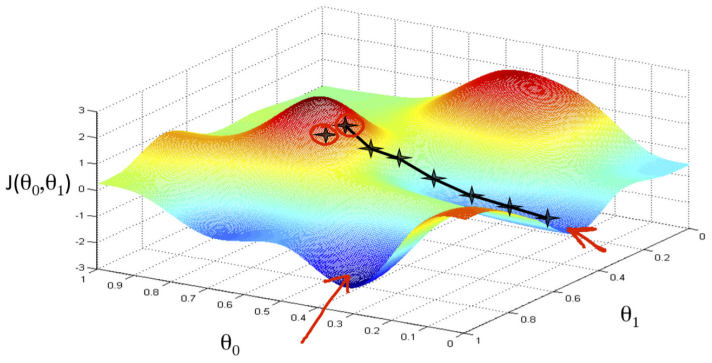


Figure 2.8: 3D Gradient Descent.

Although mini-batch gradient descent is a popular optimization algorithm for deep learning, it has some limitations and may not always guarantee good convergence.

To address these challenges, several optimization algorithms have been developed and widely used by the DL community, including Momentum [40], Nesterov accelerated gradient [41], and others. These algorithms aim to improve the convergence speed and stability of the optimization process, leading to better overall performance of the deep learning model.

2.8.2 Batch size

The batch size is a critical hyperparameter in machine learning that specifies the number of samples processed in each iteration before updating the model's internal parameters. It plays a vital role in ensuring the optimal performance of machine learning models. [42].

The batch can be considered a loop repeated through one or more examples and generates predictions. Once the forward algorithm is complete, the error is calculated by comparing predictions to the expected output. The update algorithm is used to enhance the model using this error value. The training dataset can be divided into one or more batches.

2.8.3 Epochs

An "epoch" refers to processing an entire dataset through a neural network once. To avoid overwhelming the computer, we divide the epoch into smaller batches. Since one epoch is not enough for weight updates, we use multiple epochs. The more epochs we use, the more frequently the weights are adjusted, causing the curve to shift from underfitting to the optimum to overfitting, as shown in (2.9). There is no specific number of epochs to use, but we can assume that the number is proportional to the diversity of the data.

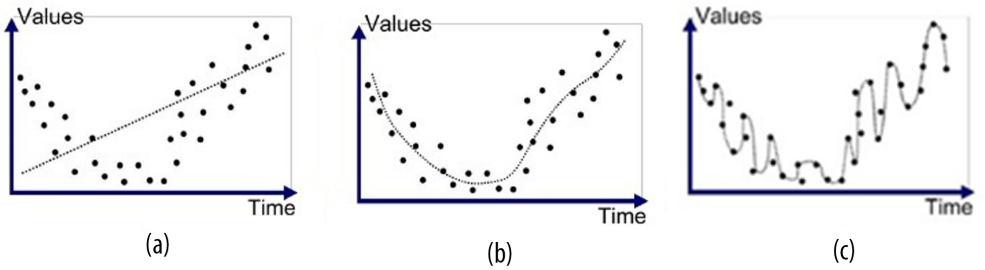


Figure 2.9: Model Performance Across Epochs: Exploring (a) Underfitting, (b) Optimum and (c) Overfitting

.

2.8.4 Learning rate

The learning rate (LR) is typically the most influential of all the hyperparameters. Its impact can be significant, affecting not only the duration of training a neural network but also whether the network converges on a local (suboptimal) or global (optimal) minimum during optimization [43].

The LR determines the step size we take toward the optimal solution during training. A higher LR value will result in faster convergence but may lead to overshooting the optimal solution. In comparison, a lower LR value will result in slower convergence but may miss the optimal solution entirely. Therefore, finding the optimal LR value is crucial for achieving high accuracy and rapid convergence in deep

learning models.

Dynamic LRs help to adjust the LR values at various stages of the training process, striking a balance between underfitting and overfitting and achieving better generalization performance.

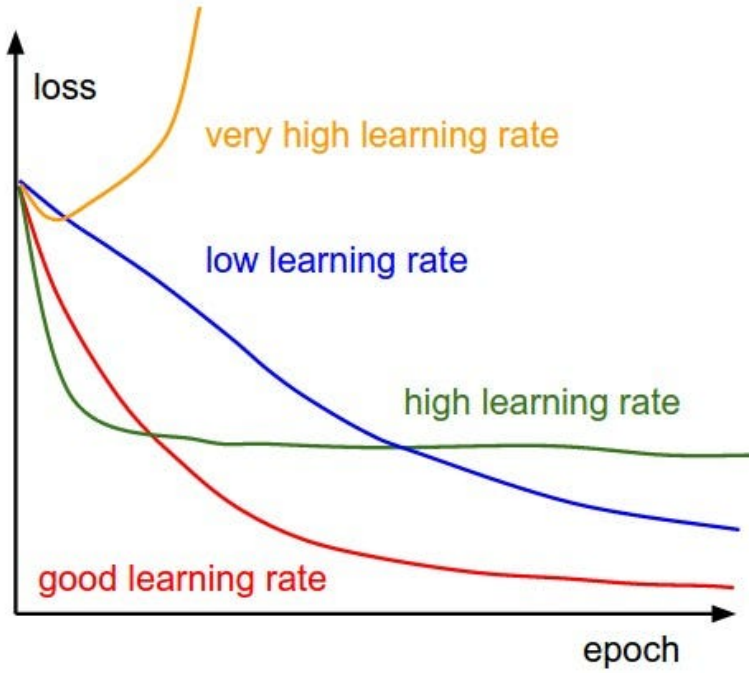


Figure 2.10: The Impact of Learning Rate on Loss during Training [44].

2.8.5 Steps per epoch

Steps per epoch refer to the number of times the model's training loop iterates to update its parameters. Each iteration involves processing a batch of data using the gradient descent technique. Typically, the number of steps per epoch is calculated as the total data points in the training set divided by the batch size. When the data has been augmented, this calculation may be multiplied by a factor of 2 or 3. However, if the training has been ongoing for a while, it may be best to stick with the original calculation method.

2.9 Feedforward and backpropagation

2.9.1 Feedforward

Feedforward, also called forward propagation, is a crucial step in neural networks where input data is processed through layers of neurons to generate an output. This process is used for making predictions and training deep learning models, forming a critical part of the overall neural network architecture.

2.9.2 Backpropagation

Backpropagation is a technique used in artificial neural networks to compute the gradients of the loss function with respect to the weights and biases of the network. This allows the network to adjust its weights and biases during training to improve performance. Backpropagation gets its name from the fact that errors in the output are propagated backward through the layers of the network. It is commonly used to train deep neural networks and is essential in a wide range of applications [45].

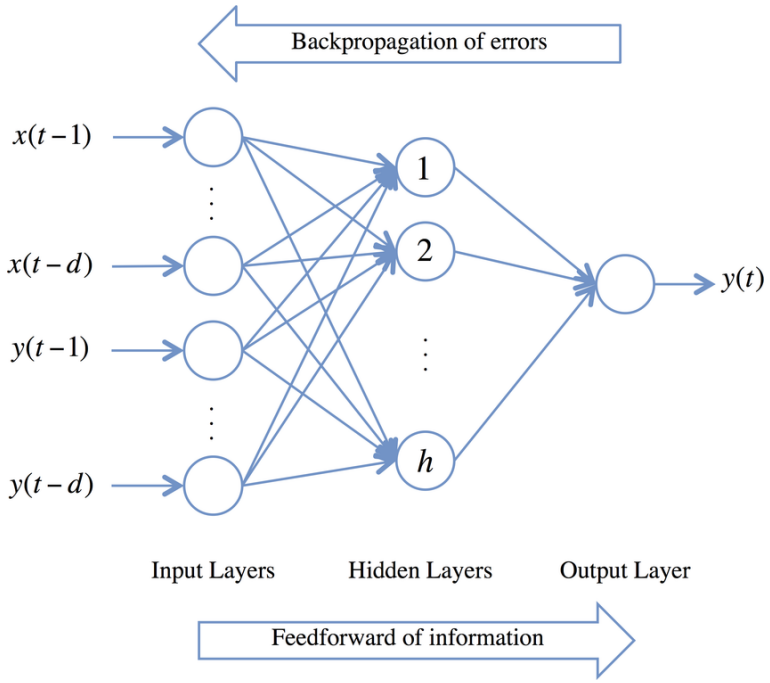


Figure 2.11: Feedforward and Backpropagation Neural Network architecture.

2.10 Data augmentation

Data augmentation is a crucial technique used to impart the desired invariance and robustness properties to a neural network, especially when there is limited availability of training samples [46].

Increased data involves applying a series of shifts to a specific set of training data to create more diverse and enhanced data. Other modification techniques include introducing fine requirements for cost function, normalizing the outputs of different layers, changing the model size and structure, changing the number of learning parameters or synaptic link pattern, and changing the training procedure. Accurate learning and tuning transfer can also be used when it is difficult to access large training data. Another approach is to focus on developing models based on more effective representations of the characteristics of training data. Increased data specifically refers to methods that seek to improve performance by changing the characteristics of the training dataset itself [47].

2.11 Convolutional neural networks

Convolutional Neural Networks (CNN) are a neural network used for processing data in multiple arrays such as images or audio spectrograms. They utilize several key ideas such as local connections, shared weights, pooling, and many layers to capture local patterns and features, learn translation-invariant features, improve computational efficiency, and learn hierarchical representations of the input data. ConvNets are highly effective for image classification, object detection, and other tasks and have revolutionized many fields by their ability to learn features from raw data [48].

When creating a CNN structure, it is important to include four basic layers: dilution layers, nonlinear layers, pool layers, and fully connected layers (FC).

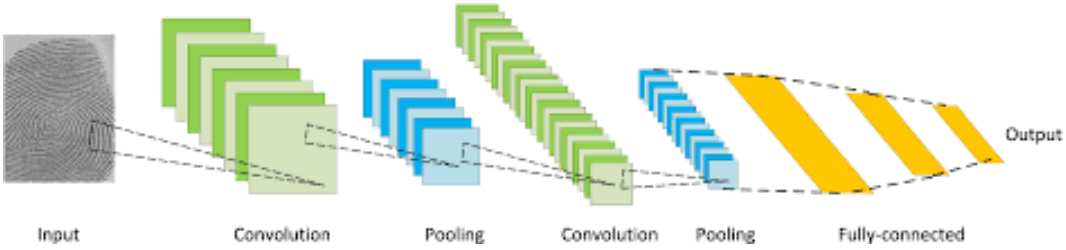


Figure 2.12: Architecture of convolutional neural networks [49].

A convolutional layer is crucial in extracting features from input images in a CNN. It achieves this using a filter or kernel to perform a dot product operation with the input image. The output of this operation is a feature map that is then passed on to a nonlinear layer, which applies an activation function, such as ReLU or Softmax, to the feature map. This enables the network to model nonlinear functions.

After the convolutional layer, it is common to include a pooling layer that reduces the size of the feature maps by applying statistical functions, such as mean or max pooling, to small neighborhoods.

Finally, the output of all previous layers is flattened and fed into a fully connected (FC) layer, where mathematical functions are applied to generate the final result. This process is illustrated in figure 2.12.

2.12 Common convolutional neural network architectures

Several commonly used convolutional neural network (CNN) architectures exist in computer vision, including LeNet-5, AlexNet, VGG-16, GoogLeNet, and ResNet. These networks have significantly contributed to the field and are often used as building blocks for many segmentation architectures [50]. in (figure 2.13) shows an overview of these architectures.

We will use the following mathematical equation to calculate the output of con-

volutional layers:

$$\left[\frac{n+2p-f}{s}+1\right] * \left[\frac{n+2p-f}{s}+1\right]$$

Where we consider a $n * n$ image as an input, a $f * f$ filter, a padding p, and a stride s.

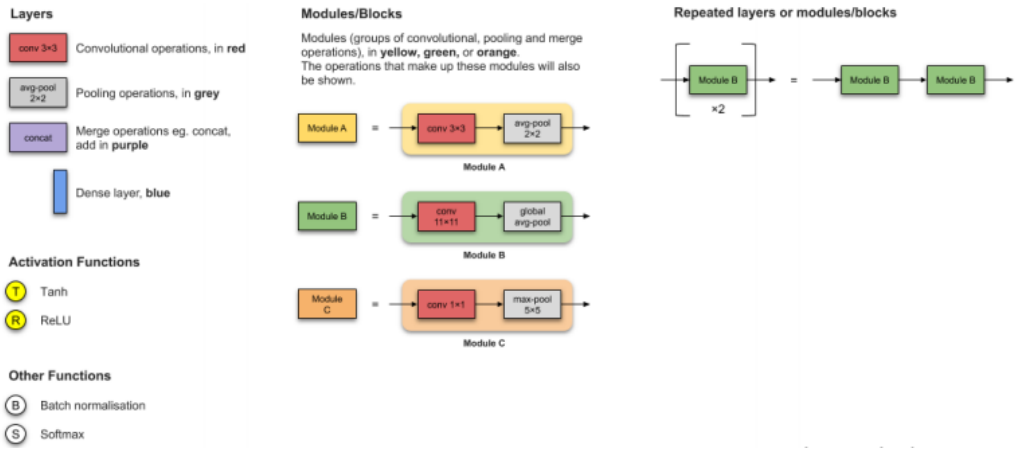


Figure 2.13: Legend used for the various architectures [51].

2.12.1 LeNet-5

LeNet-5 is a pioneering 7-level convolutional network developed by LeCun et al.in 1998 [52]. This architecture was developed to recognize handwritten numbers on scanned checks with 32x32 pixel grayscale input images, which several banks used. It is considered one of the most basic architectural designs, consisting of 61K parameters, two convolutional layers, and three fully connected layers.

This architecture has become the industry standard: convolutions with activation functions are stacked, layers are pooled, and the network is finished with one or more completely linked layers. It has the architecture represented in the following Table, see (Table 2.1).

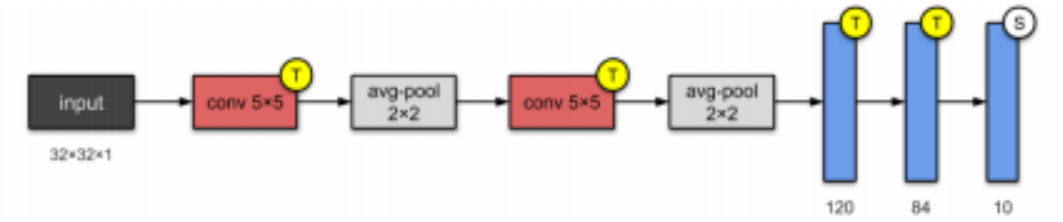


Figure 2.14: LeNet architecture from [51].

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (Grayscale)	1	32x32	-	-	-	-
1	Convolution	6	28x28	5x5	1	tanh	156
2	Average Pooling	6	14x14	2x2	2	tanh	0
3	Convolution	16	10x10	5x5	1	tanh	2416
4	Average Pooling	16	5x5	2x2	2	tanh	0
5	Convolution	120	1x1	5x5	1	tanh	48120
6	FC	-	84	-	-	tanh	10164
Output	FC	-	10	-	-	softmax	850
Total number of parameters							61,706

Table 2.1: LeNet structural details

2.12.2 AlexNet

AlexNet was a deep convolutional neural network designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, and it was submitted to the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [53].

Although the network architecture shared similarities with LeNet, it was considerably more intricate and extensive, with 62 million parameters and a higher quantity of filters employed in each layer. Furthermore, the network utilized multiple layers of convolutional operations to identify significant features from the input data.

The network is composed of five convolutional layers, max-pooling layers, Rectified Linear Units (ReLUs) as activation functions, and three fully-connected layers [54] see (Table 2.2).

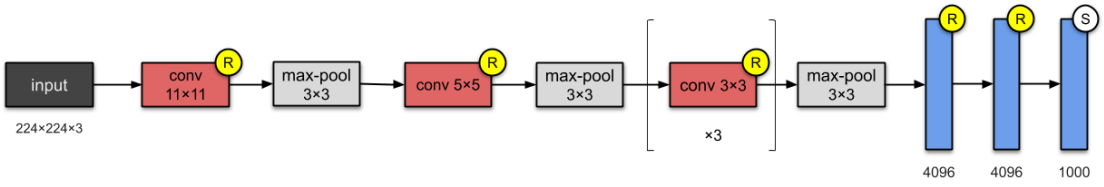


Figure 2.15: AlexNet architecture from [51].

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (RGB)	1	227x227x3	-	-	-	-
1	Convolution	96	55x55x96	11x11	4	relu	34944
	Max Pooling	96	27x27x96	3x3	2	relu	0
2	Convolution	256	27x27x256	5x5	1	relu	614656
	Max Pooling	256	13x13x256	3x3	2	relu	0
3	Convolution	384	13x13x384	3x3	1	relu	885120
4	Convolution	384	13x13x384	3x3	1	relu	1327488
5	Convolution	256	13x13x256	3x3	1	relu	884992
	Max Pooling	256	6x6x256	3x3	2	relu	0
6	FC	-	4096	-	-	relu	37752832
7	FC	-	4096	-	-	relu	16781312
Output	FC	-	1000	-	-	softmax	4097000
Total number of parameters							62,378,344

Table 2.2: AlexNet structural details

.

2.12.3 Visual Geometry Group (VGG 16,VGG19)

VGG16 and VGG19 are convolutional neural network (CNN) architectures that Karen Simonyan and Andrew Zisserman proposed from the Visual Geometry Group Lab at Oxford University. These models were introduced paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition." Both VGG16 and VGG19 were developed for large-scale image recognition.

VGG16 achieved remarkable success in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014. It secured first and second places in the image classification and localization categories. This recognition helped solidify the effectiveness of deep convolutional neural networks in computer vision tasks [55].

VGG16, as the name suggests, consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. It utilizes 3x3 convolutional filters with a stride of 1 pixel, and after each set of convolutional layers, max-pooling layers with a 2x2 filter and a stride of 2 pixels are applied.

The architecture of VGG19, an extension of VGG16, includes two additional convolutional layers. Thus, VGG19 comprises 19 layers, with similar layer configurations as VGG16.

Both VGG16 and VGG19 employ the rectified linear unit (ReLU) activation function after each layer to introduce non-linearity. They also incorporate the concept of deep learning by stacking multiple convolutional and pooling layers to learn hierarchical representations of visual features.

These VGG models played a significant role in advancing the field of deep learning and demonstrated the effectiveness of deep CNN architectures for image recognition tasks. They have since been widely used as baselines for benchmarking new architectures and have influenced subsequent developments in computer vision.

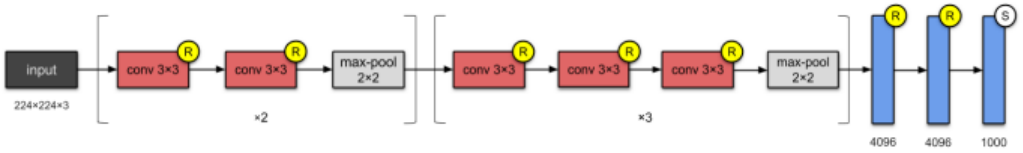


Figure 2.16: VGG-16 architecture from [51].

Layer		Feature Map	Size	Kernel Size	Stride	Activation	Parameters
Input	Image (RGB)	1	224x224x3	-	-	-	-
1	2xConvolution	64	224x224x64	3x3	1	relu	38720
	Max Pooling	64	112x122x64	2x2	2	relu	0
3	2xConvolution	128	112x122x128	3x3	1	relu	221440
	Max Pooling	128	56x56x128	2x2	2	relu	0
5	3xConvolution	256	56x56x256	3x3	1	relu	1475328
	Max Pooling	256	56x56x256	2x2	2	relu	0
7	3xConvolution	512	28x28x512	3x3	1	relu	5899776
	Max Pooling	512	14x14x512	2x2	2	relu	0
10	3xConvolution	512	14x14x512	3x3	1	relu	7079424
	Max Pooling	512	7x7x512	2x2	2	relu	0
14	FC	-	4096	-	-	relu	102764544
15	FC	-	4096	-	-	relu	16781312
Output	FC	-	1000	-	-	softmax	4097000
Total number of parameters							134,264,641

Table 2.3: VGG-16 structural details using padding = 1

.

2.12.4 ResNet

In the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) of 2015, Kaiming He and his colleagues proposed a new deep neural network architecture called ResNet (short for Residual Network) [56]. The design of ResNet introduced "skip connections" and significant batch normalization (BN). These connections also referred to as gated units or gated recurrent units, bear similarities to successful components of recent RNNs.

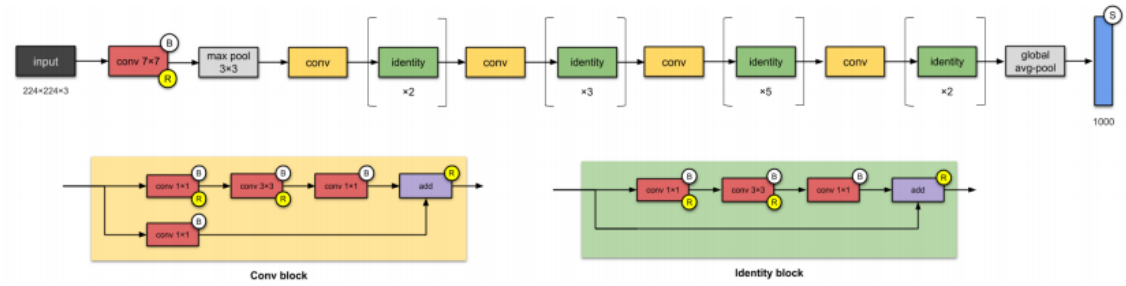


Figure 2.17: ResNet-50 architecture from [51].

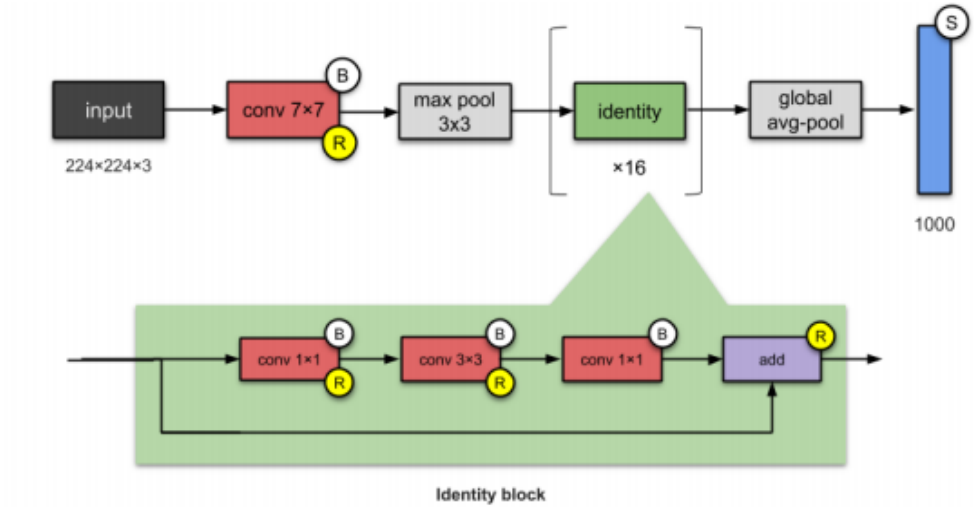


Figure 2.18: ResNet identity block from [51].

The intuition behind this approach is that by connecting the previous layer's output with the unchanged input, the subsequent layer is encouraged to learn new and distinct information not already encoded in the input. This helps to prevent redundancy and promotes the discovery of novel features and patterns. Additionally, these connections can help mitigate the vanishing gradients problem, which refers to the issue of gradients becoming extremely small as they propagate through deep neural networks, making it difficult for earlier layers to learn effectively. By providing direct connections, the gradients have a shorter path to flow through, addressing the vanishing gradients problem to some extent [54].

Layer	Input size	Output size	Filter	Parameters
Convolution1	224x224x3	112x122x64	7x7x64, stride = 2	9472
3xConvolution2 (Convolution Block + 2x(Identity block))	112x122x64	56x56x64	3x3 Max pooling, stride = 2	0
			1x1x64 3x3x64 1x1x256	214800
4xConvolution3 1x(Convolution Block) + 3x(Identity block)	56x56x64	28x28x128	1x1x128 3x3x128 1x1x512	1216000
6xConvolution4 1x(Convolution Block) + 5x(Identity block)	28x28x128	14x14x256	1x1x256 3x3x256 1x1x1024	7088128
3xConvolution5 1x(Convolution Block) + 2x(Identity block)	14x14x256	7x7x512	1x1x512 3x3x512 1x1x2048	14953472
FC	7x7x512	1x1x1000	Average pooling, 1000-d FC, Softmax	2049000
Total number of parameters				25,636,712

Table 2.4: ResNet structural details.

2.12.5 GoogleNet

In 2014, Google researchers introduced GoogLeNet, a deep convolutional neural network with Inception architecture. It achieved remarkable results on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dataset, a commonly used benchmark for object recognition tasks. Since then, GoogLeNet has become widely recognized and studied in computer vision. Its architecture was first described

in a research paper "Going Deeper with Convolutions" published the same year [57].

GoogLeNet uses the Inception module to apply multiple operations on the same input map simultaneously. This module runs multiple kernels of different sizes in parallel and concatenates their outputs to produce a single output. By using filters of different sizes on the same level, the network can become "wider" instead of "deeper", which was the intention of the module's creators. The architecture was designed for computational efficiency and practicality, enabling it to be run on devices with limited processing power and memory. The network has 22 layers (or 27 layers when including pooling layers) if we only count layers with parameters.

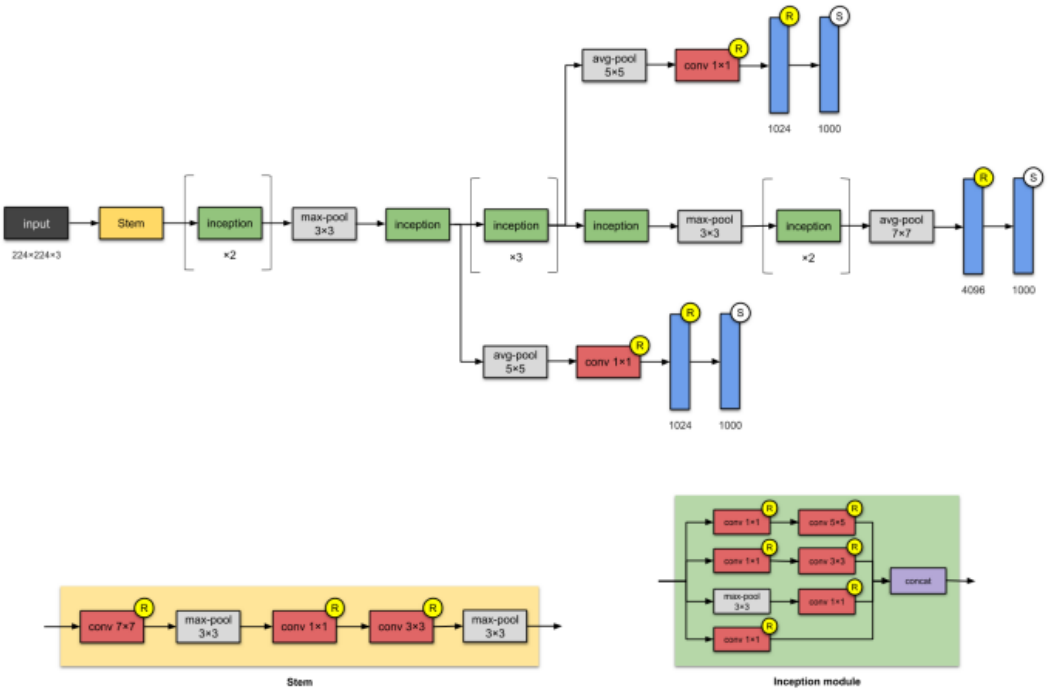


Figure 2.19: GoogleNet architecture (Inception V1) from [51].

The GoogLeNet architecture consists of approximately 100 layers, but the specific number may vary depending on the machine learning infrastructure used. Unlike other popular networks like VGGNet and AlexNet, GoogLeNet has a different structure, including a 1×1 Convolution layer at the center of the network and global average pooling at the end instead of fully connected layers. With around 6.8 million parameters (excluding auxiliary layers), GoogLeNet has significantly fewer parameters than AlexNet (9 times less) and VGG-16 (20 times less), making it a more efficient network in terms of memory and processing power.

Type	Patch size / Stride	Output size	Depth	#1x1	#3x3 Reduce	#3x3	#5x5 Reduce	#5x5	Pool Proj	Parameters
Convolution	7x7/2	112x112x64	1	-	-	-	-	-	-	2.7K
Max Pooling	3x3/2	56x56x64	0	-	-	-	-	-	-	-
Convolution	3x3/1	56x56x192	2	-	64	192	-	-	-	112K
Max Pooling	3x3/2	28x28x192	0	-	-	-	-	-	-	-
Inception (3a)	-	28x28x256	2	64	96	128	16	32	32	159K
Inception (3b)	-	28x28x480	2	128	128	192	32	96	64	380K
Max Pooling	3x3/2	14x14x480	0	-	-	-	-	-	-	-
Inception (4a)	-	14x14x512	2	192	96	208	16	48	64	364K
Inception (4b)	-	14x14x512	2	160	112	224	24	64	64	437K
Inception (4c)	-	14x14x512	2	128	128	256	24	64	64	463K
Inception (4d)	-	14x14x528	2	112	144	288	32	64	64	580K
Inception (4e)	-	14x14x832	2	256	160	320	32	128	128	840K
Max Pooling	3x3/2	7x7x832	0	-	-	-	-	-	-	-
Inception (5a)	-	7x7x832	2	256	160	320	32	128	128	1072K
Inception (5b)	-	7x7x1024	2	384	192	384	48	128	128	1388K
Average Pooling	7x7/1	1x1x1024	0	-	-	-	-	-	-	-
Dropout (40%)	-	1x1x1024	0	-	-	-	-	-	-	-
Linear	-	1x1x1000	1	-	-	-	-	-	-	1000K
Softmax	-	1x1x1000	0	-	-	-	-	-	-	-
Total number of parameters										6,8M

Table 2.5: GoogleNet structural details.

2.12.6 MobileNet

MobileNet is a class of efficient neural network architectures designed for mobile and embedded vision applications. It was developed by Google researchers in 2017 [58] as a part of the TensorFlow library. MobileNets employ depthwise separable convolutions, which decompose the standard convolution operation into two separate steps:

- **Depthwise convolution:** is a spatial convolution performed independently on each input channel. The output channels are then concatenated to form the output feature map. This reduces the number of parameters and computation required compared to a standard convolution. The computational cost of depthwise convolution is proportional to the depth of the input feature map M, the square of the kernel size Dk^2, and the square of the output feature map size Df^2.

- **Pointwise convolution:** Is a 1x1 convolution applied to the output of depthwise convolution. It uses a kernel size of 1x1 to perform a linear combination of the output channels. This further reduces the number of parameters and computations required. The computational cost of pointwise convolution is proportional to the number of input and output channels X.

This approach significantly reduces the number of parameters in the network and leads to lightweight models that can be run on resource-constrained devices. MobileNets have been widely used in various computer vision tasks, such as image classification, object detection, and semantic segmentation.

The MobileNet architecture consists of a total of 28 layers, including both depthwise and pointwise convolutions. Adjusting the width multiplier hyperparameter can reduce the parameters in a standard MobileNet to 4.2 million. The input image size is $224 \times 224 \times 3$. The detailed architecture of a MobileNet is given below:

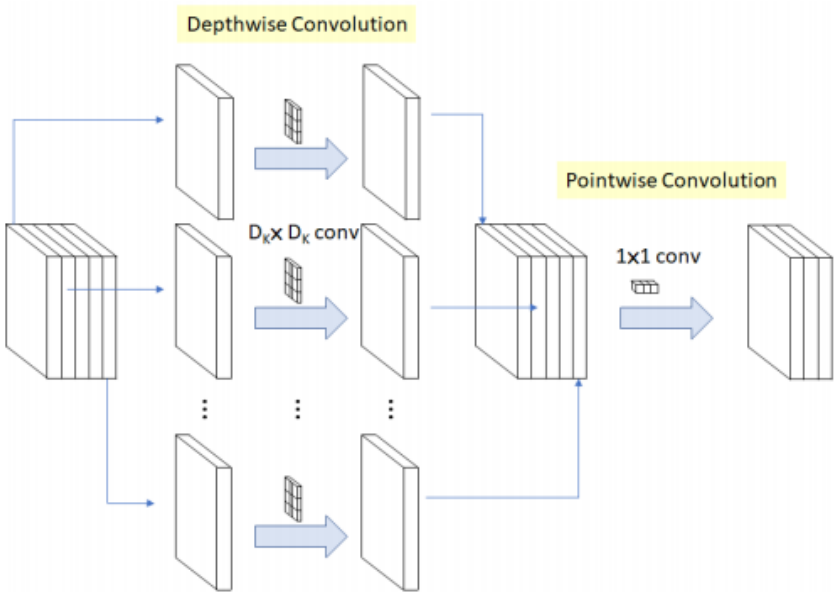


Figure 2.20: MobileNet architecture from [59].

Layer	Input size	Stride	Filter	Parameters
Convolution1	224x224x3	2	3x3x3x32	992
Convolution_dw1	112x112x32	1	3x3x32 dw	416
Convolution_pw1	112x112x32	1	1x1x32x64	2304
Convolution_dw2	112x112x32	2	3x3x64 dw	832
Convolution_pw2	56x56x64	1	1x1x46x128	8704
Convolution_dw3	56x56x128	1	3x3x128 dw	1664
Convolution_pw3	56x56x128	1	1x1x128x128	16896
Convolution_dw4	56x56x128	2	3x3x128 dw	1664
Convolution_pw4	56x56x128	1	1x1x128x256	33792
Convolution_dw5	56x56x256	1	3x3x256 dw	3328
Convolution_pw5	56x56x256	1	1x1x256x256	66560
Convolution_dw6	56x56x256	2	3x3x256 dw	3328
Convolution_pw6	14x14x256	1	1x1x256x512	133120
5xConvolution_dw7	14x14x512	1	3x3x512 dw	33280
5xConvolution_pw7	14x14x512	1	1x1x512x512	1320960
Convolution_dw8	14x14x512	2	3x3x512 dw	6656
Convolution_pw8	7x7x512	1	1x1x512x1024	528384
Convolution_dw9	7x7x1024	2	3x3x1024 dw	13312
Convolution_pw10	7x7x1024	1	1x1x1024x1024	1052672
Average pooling	7x7x1024	1	7x7x1024	0
FC	7x7x1024	1	1024x1000	1025000
Softmax	1x1x1000	1	-	0
Total number of parameters				4,253,864

Table 2.6: MobileNet structural details.

Conclusion

We have covered the most prevalent deep learning concepts in the previous chapter. In the next chapter, we will delve into deep learning applications in computer vision, with a particular focus on semantic image segmentation and object detection.

Chapter 3

Semantic Segmentation and Object Detection

Introduction

Computer Vision is one of the most important fields in artificial intelligence and computer science, which involves teaching computers to interpret and understand images and videos.

In this chapter, we will specifically cover two important tasks within computer vision: segmentation and object detection. Additionally, we will explore various techniques and approaches used to solve these two important computer vision tasks in the context of deep learning. We will discuss popular architectures such as UNet and its variants, DeepLab, YOLO, RCNN, and SSD, and examine their respective strengths and weaknesses. By the end of this chapter, readers should have a clear understanding of these tasks and the techniques used to solve them in the context of deep learning in computer vision.

3.1 semantic segmentation

Semantic segmentation is a method used in computer vision to assign a label to each pixel in an image, with the aim of dividing the image into meaningful and semantically consistent regions[60].

This involves assigning labels to each pixel that correspond to specific objects or regions of interest. Unlike object detection or recognition, which focuses on detecting and classifying objects in an image, semantic segmentation produces a segmentation map that assigns a label to each pixel in the image. This technique has various applications, such as in autonomous driving, medical image analysis, and scene understanding for robotics[61]. To achieve semantic segmentation, deep learning techniques like convolutional neural networks (CNNs) or fully convolutional neural networks (FCNs) are used, which learn to perform pixel-wise classification by predicting the probability distribution over classes for each pixel location [62].

CNNs are ideal for semantic segmentation because they can effectively identify local features in an image by applying convolutional filters that extract relevant

information at various spatial scales. Meanwhile, FCNs are a specific type of CNNs tailored for semantic segmentation that incorporate both convolutional and deconvolutional layers to accurately predict segmentation maps based on input images.

3.2 Techniques and Approaches for Semantic Segmentation in Deep Learning

U-Net and its variations,including Attention U-Net and Residual Attention U-Net, as well as DeepLab, are examples of deep learning-based semantic segmentation methods.

3.2.1 U-Net

U-Net is a popular deep learning architecture for image segmentation tasks, particularly in the biomedical field. It was introduced in a 2015 paper by Olaf Ronneberger, Philipp Fischer, and Thomas Brox [46]. the U-Net architecture 3.1 relies on data augmentation to learn more effectively from the available labeled data. The use of data augmentation techniques, such as rotation, scaling, and flipping of the input images, helps to increase the diversity of the training data and improve the robustness of the model to variations in the input.

The U-Net architecture is designed for image segmentation tasks, where the goal is to classify each pixel in an image into one of several categories. It consists of two paths the first path is called the contracting path or encoder or analysis path, The second path is called the expansion path or a decoder or synthesis path [63].

The encoder path consists of a series of convolutional and max-pooling layers, which progressively reduce the spatial dimensions of the input image while increasing the number of channels. This process allows the network to capture high-level features of the image, such as edges, corners, and textures.

The decoder path uses a series of up-convolutional and concatenation layers to increase the spatial resolution of the feature maps while reducing the number of channels. The skip connections between the encoder and decoder paths allow the network to recover detailed information from earlier layers and improve the segmentation accuracy.

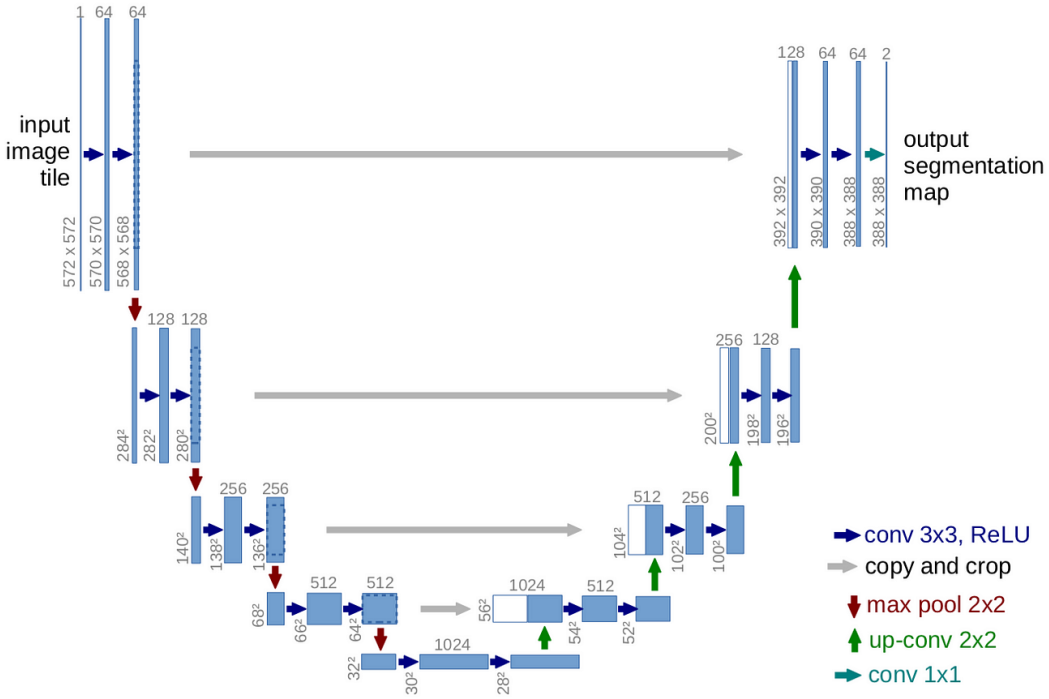


Figure 3.1: U-Net architecture

This describes the basic components of the U-Net architecture for image segmentation:

- **Input image tile:** the U-Net model takes an input image tile as its input, which is typically a small region of an image.
- **Convolution with 3x3 filter and ReLU activation:** this operation applies a convolutional filter with a size of 3x3 to the input image tile, followed by a rectified linear unit (ReLU) activation function. This helps to extract features from the input image.
- **Copy and crop:** after each convolutional layer in the contracting path, the feature maps are copied and concatenated with the corresponding feature maps in the expanding path. Before concatenation, the feature maps in the expanding path are cropped to match the size of the corresponding feature maps in the contracting path.
- **Max pooling with 2x2 filter:** this operation reduces the spatial resolution of the feature maps while preserving their important features. It is used in the contracting path to downsample the feature maps.
- **Up-convolution with 2x2 filter:** this operation increases the spatial resolution of the feature maps while reducing their number of channels. It is used in the expanding path to upsample the feature maps.
- **Convolution with 1x1 filter:** this operation is used to perform a linear combination of the channels in the feature maps, reducing their dimensionality and increasing their representational power.

- **Output segmentation map:** the final output of the U-Net model is a segmentation map, which assigns a label to each pixel in the input image tile based on its predicted class.

3.2.2 Attention U-Net

Attention U-Net is a variant of the U-Net architecture for image segmentation that was introduced in a 2018 paper by Ozan Oktay, et al[64],The Attention U-Net architecture is designed for image segmentation and incorporates an attention mechanism that allows the model to focus on relevant regions of the input image while ignoring irrelevant regions. The model takes in an input image and applies a series of operations including a 3x3 convolution followed by a ReLU activation, upsampling by a factor of 2, and max pooling. The output is then fed into a skip connection.

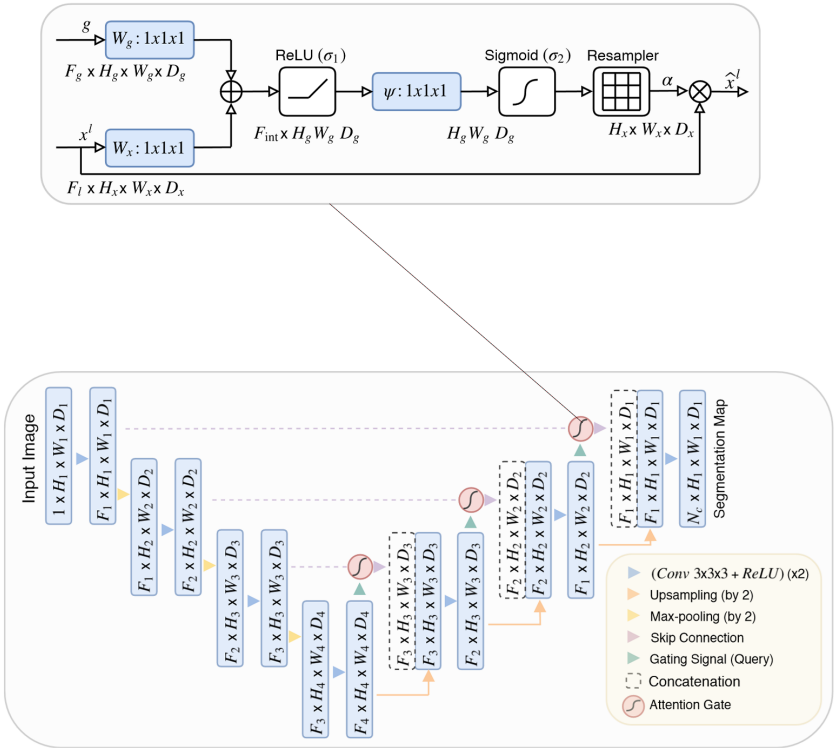


Figure 3.2: Attention U-Net architecture from [64]

The attention mechanism in the Attention U-Net is based on a gating mechanism that assigns weights to different regions of the feature maps. The attention maps are computed by applying a softmax function to the output of a convolutional layer, which is used to scale the feature maps. The attention gate requires two inputs: x, which is obtained from the output of the previous layer, and g, which is obtained from the next lowest tier of the network and has lower dimensions but better feature representation.

The vectors x and g are processed and added element-by-element. The resulting vector is then passed through a ReLU activation layer and a 1x1 convolution layer,

reducing the dimensions to 1x32x32. A sigmoid layer is used to scale the resulting vector between 0 and 1, producing the attention coefficients or weights. The attention coefficients are upsampled to the original dimensions of the x vector using trilinear interpolation.

the original x vector is multiplied element-by-element with the attention coefficients, scaling the vector according to relevance, before being passed along normally in the skip connection. By incorporating the attention mechanism into the U-Net architecture, the Attention U-Net can improve the accuracy of image segmentation, particularly in cases where there are complex structures or background noise in the input image.

3.2.3 Attention Residual U-Net

Residual Attention U-Net is an extension of the Attention U-Net architecture. It incorporates standard convolutional blocks that include one or more convolutional operations followed by ReLU activation and an optional batch normalization layer.

The output of these blocks is then combined with the input through residual connections. These residual connections enable the model to learn the residual information between the input and output during training, resulting in improved information flow and performance in image segmentation tasks.

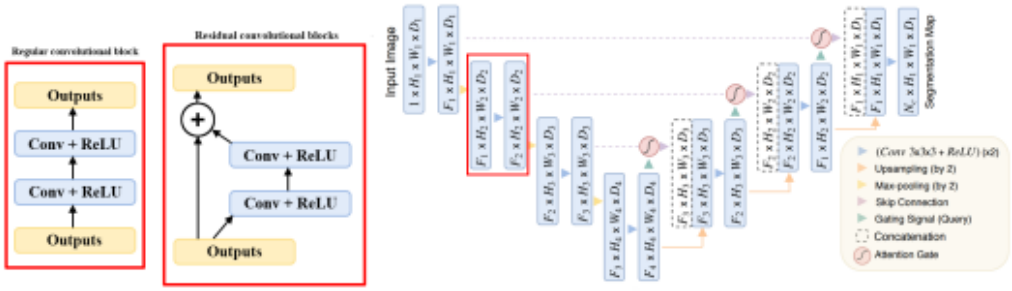


Figure 3.3: Attention Residual U-Net architecture

3.2.4 DeepLab

Chen and colleagues developed two of the most commonly used methods for image segmentation, DeepLabv1 [65] and DeepLabv2 [66].

The latter method has three key features that set it apart. Firstly, it uses dilated convolution to address the network's reduced resolution caused by max-pooling and striding. Secondly, it incorporates ASPP, which employs Atrous convolution with varying dilation to capture information about objects at multiple scales. Finally, the method combines deep CNNs and probabilistic graphical models to improve object boundary localization.

The researchers subsequently introduced DeepLabv3 [67], which comprises cascaded and parallel dilated convolution modules. The ASPP module gathers the

parallel convolution modules, including a 1x1 convolution and batch normalization. To create the final output with logits for each pixel, all outputs are concatenated and passed through another 1x1 convolution.

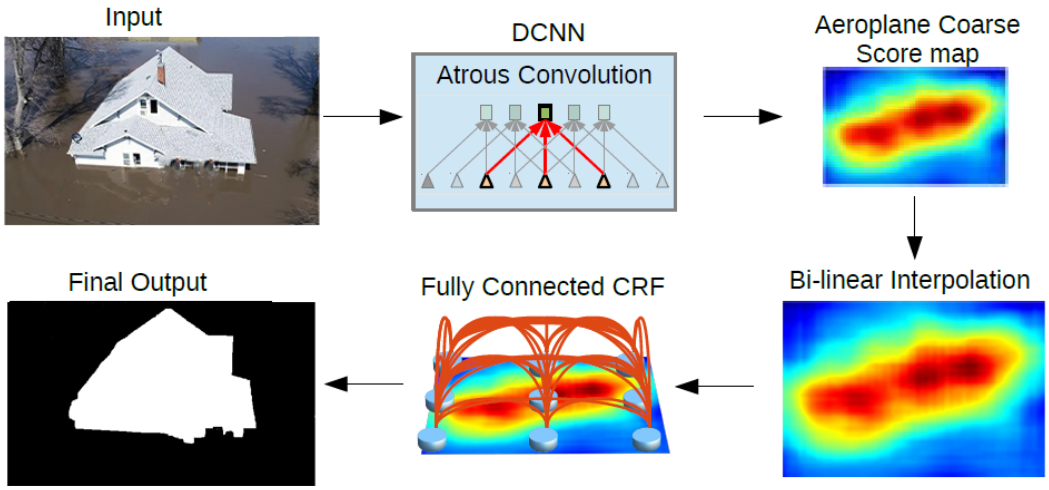


Figure 3.4: The DeepLab model

Deeplabv3+ was released in 2018 [68],The architecture of this method uses an encoder-decoder architecture (figure 3.5) that incorporates atrous separable convolution, which consists of a depthwise convolution (a spatial convolution for each input channel) and a pointwise convolution (a 1x1 convolution with the depthwise convolution as input).

The DeepLabv3 framework is used as the encoder, and the model employs a modified Xception [69] backbone with additional layers, dilated depth-wise separable convolutions instead of max pooling, and batch normalization (BN) instead of max pooling.

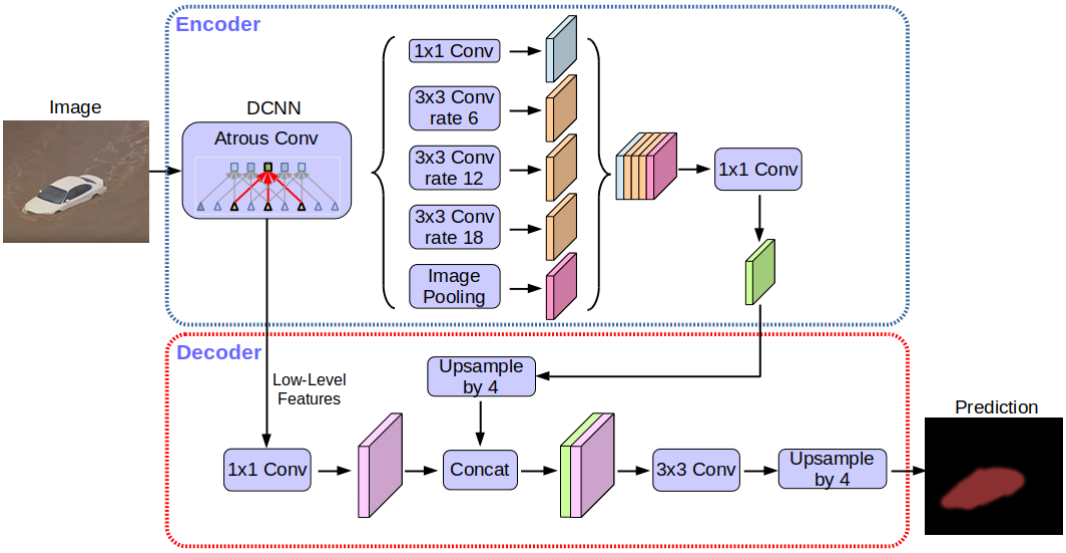


Figure 3.5: The DeepLabv3+ model from [68]

3.3 Performance Metrics for Semantic Segmentation

Once a machine learning model has been developed for semantic segmentation, the next crucial step is to evaluate its predictive performance. To achieve this, it is necessary to partition the data into training and validation sets to compute appropriate metrics. These metrics include:

3.3.1 Intersection-Over-Union

Commonly used loss functions in DNNs, such as softmax loss, are suitable for traditional classification problems where accuracy is the primary objective. However, in the case of image segmentation, the foreground and background classes are highly imbalanced. Hence, the IoU (Intersection-Over-Union) method is commonly used to evaluate the performance of image segmentation techniques. This method calculates the IoU for each semantic class before averaging over all classes and is a typical semantic image segmentation assessment metric [70].

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

IoU is defined as follows:

$$IOU = \frac{True\ Positive}{(True\ Positive + False\ Positive + False\ Negative)}.$$

The predictions are accumulated in a confusion matrix, weighted by a variable, and the metric is then calculated.

3.3.2 Accuracy

This metric measures the proportion of correct predictions made by the model. It is calculated by dividing the number of correct predictions by the total number of predictions. The formal definition of accuracy is as follows:

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

The following formula can be used to calculate accuracy if we had positive and negative numbers in a binary classification:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

3.3.3 Precision

Precision is a measure of the accuracy of positive predictions. It indicates how well the minority class has been identified. Precision is calculated by dividing the number of correctly predicted positive cases by the total number of positive cases expected. The formal definition of Precision is as follows:

$$Precision = \frac{True\ Positives}{False\ Positives + True\ Positives}$$

3.3.4 Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive samples that are correctly classified as positive. It is calculated by dividing the number of correctly classified positive samples by the total number of actual positive samples. Unlike precision, recall takes into account the positive samples that were missed and gives an indication of how well the model is able to cover the positive class. The recall score ranges from 0 to 1, with 1 indicating that all positive samples were correctly classified. The formal definition of Recall is as follows:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

3.3.5 F-Measure (F1 Score)

The F-measure, also known as the F1 score, is a metric that combines precision and recalls into a single score to provide a comprehensive evaluation of the model's performance. It is particularly useful in situations where both precision and recall are equally important.

The F-measure is calculated as the harmonic mean of precision and recall, which is expressed mathematically as follows:

$$F1\,Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

An F1 score of 1 means that both the precision and recall are at their maximum values, indicating that the model has made no false positives or false negatives. On the other hand, a score of 0 indicates that the model has made no correct predictions at all [71].

3.3.6 Jaccard similarity coefficient

The Jaccard similarity coefficient is a versatile and straightforward formula that can be used in different contexts, such as image segmentation. It is a useful metric to evaluate the quality of image segmentation, by measuring how similar the segmentation results are to the ground truth. The formula of the Jaccard similarity coefficient involves the intersection and union of the segmentation result S and the ground truth G.

$$E = \frac{A(G \cap S)}{A(G \cup S)}$$

The Jaccard similarity coefficient equation includes an operation of counting quantity denoted by A(x). In this equation, the numerator represents the number of matching pixels or true positives, which are the pixels correctly identified as belonging to the target object or class [72].

3.4 Object Detection

Object detection is a highly dynamic research field within computer vision that involves two primary tasks: object classification, which entails identifying and labeling

every object in an image, and object localization, which involves identifying each object and drawing a bounding box around it[73].

Recent advances in deep learning have significantly improved the accuracy and efficiency of object detection systems. The development of CNNs has enabled the creation of more powerful and accurate models that can process large amounts of data quickly. One popular approach in object detection is YOLO and R-CNN. These models have demonstrated impressive results in object detection tasks and have been widely adopted in various computer vision applications.

Object detection is a field with numerous applications, ranging from surveillance and security to autonomous driving with UAVs. It also has a critical role in object recognition and scene comprehension, which are fundamental tasks in computer vision. With the continuous advancement of computer vision, object detection systems are expected to become more precise and efficient, creating new opportunities for their utilization in various fields.

3.5 Techniques and Approaches for Object Detection in Deep Learning

here are some techniques and approaches used specifically in object detection:

3.5.1 You Only Look Once

YOLO is a popular object detection architecture in computer vision that was first introduced in 2016 by Joseph Redmon et al [74].

YOLO uses a single neural network to predict bounding boxes and class probabilities directly from full images, allowing for real-time object detection with high accuracy. This makes it a popular choice for applications such as surveillance, self-driving cars, and robotics. The YOLO architecture has undergone several improvements over the years, including the use of anchor boxes, batch normalization, and feature pyramid networks, making it one of the most widely used and well-regarded object detection algorithms.

There are several factors that have contributed to YOLO's success in the competition. These include its rapid processing speed, high level of detection accuracy, strong ability to generalize well across different data sets, and the fact that it is open-source and freely available to the community [75].

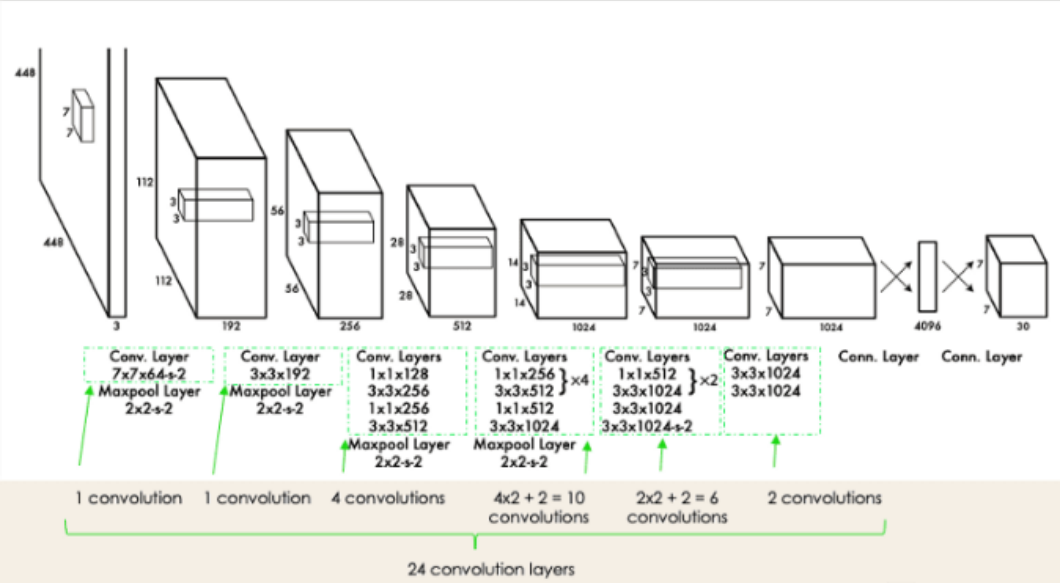


Figure 3.6: architecture of YOLO [74].

The detection network in question consists of 24 convolutional layers and 2 fully connected layers. In between the convolutional layers, there are 1x1 convolutional layers that help to reduce the feature space from the preceding layers. The network is pretrained on the ImageNet classification task using half the resolution (224x224 input image). After pretraining, the resolution is doubled for detection.

The following versions of YOLO have been released up to this point:

- **YOLOv1:** Released in 2016, it was the original version of YOLO and used a single neural network to perform object detection.
- **YOLOv2:** Released in 2017, it improved the detection accuracy and speed of YOLOv1 by using a more complex neural network architecture and implementing batch normalization [76].
- **YOLOv3:** Released in 2018, it further improved the detection accuracy and speed of YOLOv2 by using a feature pyramid network and performing multi-scale predictions [77].
- **YOLOv4:** Released in 2020, it introduced several new techniques such as spatial pyramid pooling, Mish activation function, and CIoU loss function, which significantly improved the accuracy and speed of YOLO [78].
- **YOLOv5:**Released in 2020, it is a lighter and faster of YOLOv4 that uses a different neural network architecture called the **"Scaled-YOLOv4"** and achieves state-of-the-art performance on several object detection benchmarks .

In addition to YOLOv5, newer versions have also been released, including **YOLOv6**, **YOLOv7**, and **YOLOv8**. These newer versions are focused on improving the speed and accuracy of object detection in different contexts.

3.5.2 Region-based Convolutional Neural Networks

The R-CNN It is an object detection algorithm that was introduced in 2014 by Ross Girshick, et al [79].usually referred to as R-CNNs, which is short for region-based CNN, The R-CNN family expanded to include Fast-RCNN and Faster-RCN, respectively [31].

R-CNN, as the fundamental region-based architecture, lays the groundwork for comprehending the inner workings of various other object-recognition algorithms. Although it is considered a basic approach, it has been highly influential in the field of object detection and localization by effectively applying convolutional neural networks. Its success and early adoption played a crucial role in advancing the development of more sophisticated detection algorithms that followed [31].

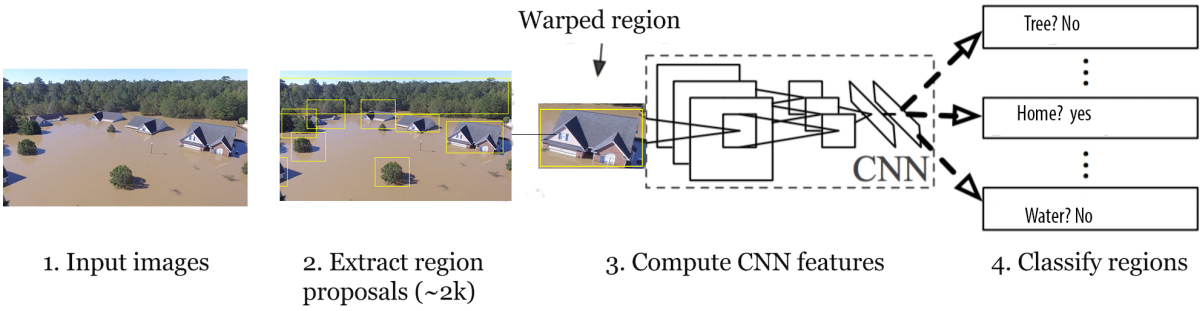


Figure 3.7: R-CNN architecture

Fast R-CNN It was introduced in 2015 by Ross Girshick at Microsoft Research [80], is a modification of the R-CNN object detection method that aims to improve detection speed while also increasing detection accuracy. It proposes a different approach to the region proposal module by applying the CNN feature extractor to the entire input image first and then proposing regions, which reduces the number of ConvNets needed to process the input [31].

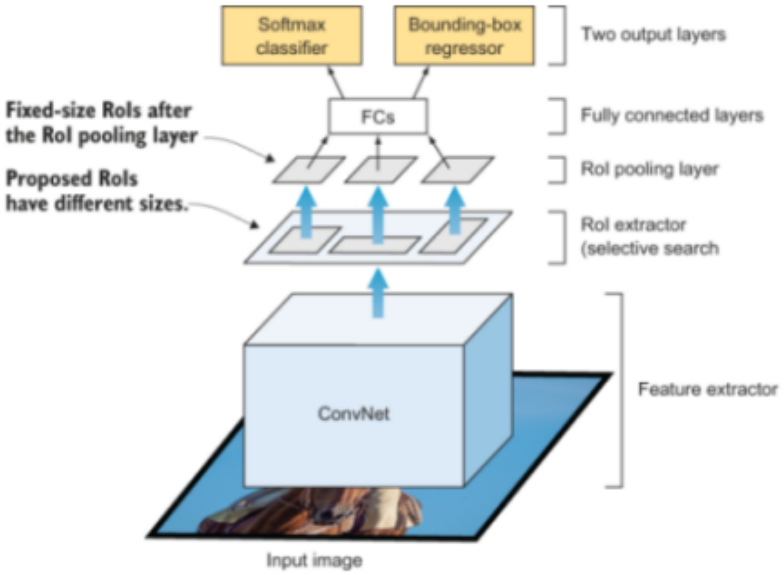


Figure 3.8: Fast CNN architecture [31].

The input image is first inserted into the algorithm, then the region proposals are created using an algorithm such as selective search. Next, features are extracted from each region proposal using a pre-trained CNN, followed by the classification of each region's proposal into one of the categories of objects. Finally, the boundary box retreat is made to revise the boxes to identify detected objects.

Faster R-CNN It was proposed in 2015 by Shaoqing Ren et al [81]. is a further modification of the R-CNN family of object detection methods that improve upon the region proposal mechanism of Fast R-CNN. Faster R-CNN introduces a Region Proposal Network (RPN) that shares the same features as the object detection network and predicts object proposals, eliminating the need for a separate proposal generation algorithm. The RPN enables Faster R-CNN to generate region proposals faster and with higher accuracy than Fast R-CNN.

Additionally, Faster R-CNN replaces the Selective Search algorithm used in R-CNN and Fast R-CNN with the RPN-based proposals, leading to even faster detection times. This approach achieved state-of-the-art results on several benchmark datasets and has become a widely used method in the field of object detection [31].

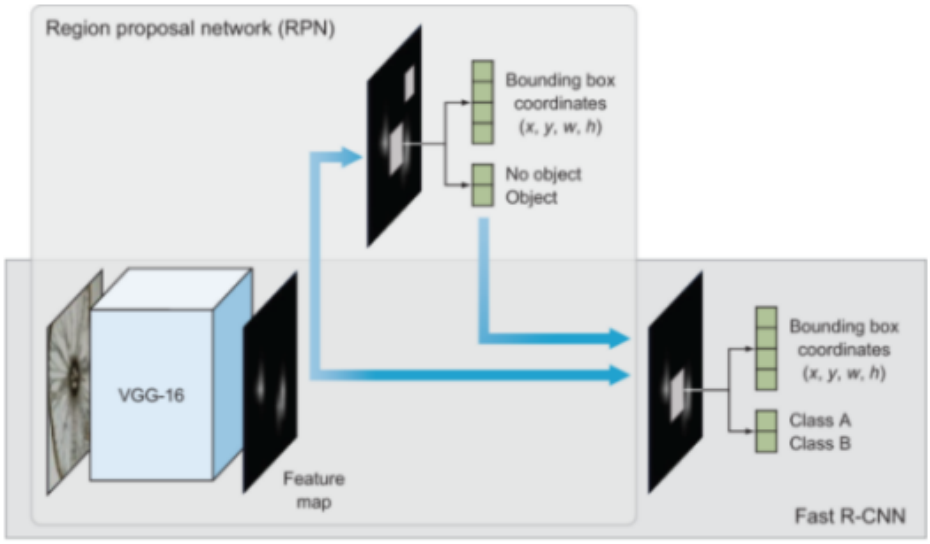


Figure 3.9: Faster CNN architecture [31].

3.5.3 Single Shot Detector

In 2016, the SSD model was introduced as a one-stage object detection algorithm [82]. Unlike two-stage detectors such as R-CNN, SSD directly predicts the object bounding boxes and class labels without the need for a separate region proposal step. SSD employs a series of convolutional layers with varying feature map resolutions to detect objects at multiple scales and aspect ratios. Additionally, the algorithm uses default bounding boxes, known as anchor boxes, at each feature map location to improve object localization. These techniques enable SSD to achieve high accuracy in object detection tasks.

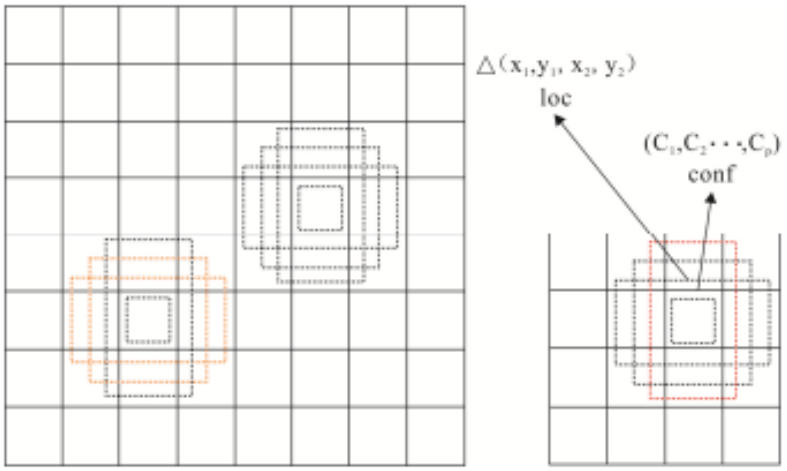


Figure 3.10: architecture SSD [83].

3.6 Performance Metrics for Object detection

Performance metrics play a vital role in assessing the accuracy and effectiveness of object detection algorithms. These metrics provide quantitative measures to evaluate the performance of the models and compare them against each other. Here are some commonly used performance metrics for object detection:

3.6.1 Average Precision

Average Precision calculates the average precision values across different levels of recall. It considers the precision and recall trade-off and provides a single numerical value to represent the overall performance of the model.The formal definition of Average Precision is as follows:

$$AveragePrecision = \frac{1}{n}\sum_{i=1}^{n} PrecisionatRecall_i$$

3.6.2 Mean Average Precision

Mean Average Precision calculates the average precision values for multiple object classes and computes the mean across all classes. It is widely used as a standard metric for object detection evaluation.The formal definition of Mean Average Precision is as follows:

$$MeanAveragePrecision = \frac{1}{m}\sum_{j=1}^{m} AveragePrecisionforclass_j$$

3.6.3 Intersection over Union

IoU measures the spatial overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the ratio of the intersection area to the union area of the two bounding boxes. A high IoU value indicates a better localization accuracy.The formal definition of IoU is as follows:

$$IoU = \frac{AreaofIntersection}{AreaofUnion}$$

3.6.4 Precision-Recall Curve

The precision-recall curve plots the precision values against different levels of recall. It provides a visual representation of the trade-off between precision and recall, allowing for an analysis of the model's performance at different operating points, which is expressed mathematically as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

3.6.5 F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, giving an overall assessment of the model's performance, which is expressed mathematically as follows:

$$F1Score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

3.6.6 Mean Average Precision at different IoU thresholds

This metric computes the average precision at specific IoU thresholds, such as 0.5, 0.75, or others. It provides insights into the model's performance at different levels of localization accuracy, which is expressed mathematically as follows:

$$MeanAveragePrecisionatIoUthreshold_x = \frac{1}{k}\sum_{j=1}^{k} AveragePrecisionatIoUthreshold_x pourclasse_j$$

Conclusion

This chapter covered the fundamental concepts and techniques for semantic segmentation and object detection in deep learning. We looked at various popular models such as UNet, DeepLab, Mask R-CNN, YOLO, and SSD, which have made significant contributions to computer vision tasks. With the rapid advancement of deep learning, we can expect even more sophisticated models and techniques to be developed in the future, making computer vision applications more powerful and accurate than ever before.

Chapter 4

Experiments and results

Introduction

In this chapter, we will showcase the outcomes of our study focusing on semantic segmentation and object detection using several models such as UNet, DeepLab, and YOLO. The chapter will contain a problem definition, dataset description, and an explanation of the data augmentation techniques employed to enhance the available data. Additionally, we will delve into the construction of the training methods and the conducted experiments. The results will be presented and compared, with evaluation using evaluation metrics. Lastly, we will briefly overview the hardware infrastructure and software tools used in our deep learning study.

4.1 Problem definition

The problem we focus on in this study revolves around the analysis of images captured by drones in areas impacted by floods. We have two primary objectives. Firstly, we aim to identify and categorize objects present in the images with precision, including vehicles, houses, trees, etc., and discern their status amidst the flooding using semantic segmentation. The second objective is to assess the extent of homes affected by the floods through object detection.

To achieve these goals, we rely on leveraging deep learning models such as UNet, DeepLab and YOLOv8n. These models provide us with the ability to analyse images accurately and efficiently, thereby enhancing the ability to assess damage and improve the accuracy of flood disaster analysis.

4.2 Experiments on semantic segmentation

This section will introduce the dataset used, outline the chosen backbone architecture, and detail the outcomes achieved through the application of the U-Net and DeepLab deep learning models on the FloodNet dataset.

4.2.1 Dataset

4.2.1.1 FloodNet

FloodNet offers high-resolution images obtained from low-altitude sources, distinguishing it from satellite imagery taken from higher altitudes that may encounter obstructions such as clouds and smoke. The unique characteristics of FloodNet's images result in clearer scenes, which can greatly assist deep-learning models in making more precise decisions about post-disaster damage assessment. By providing detailed and unobstructed visual data, FloodNet enhances the accuracy and effectiveness of deep learning algorithms in analyzing and assessing the extent of damages following a disaster. The data was collected with a small UAS platform, DJI Mavic Pro quadcopters, after Hurricane Harvey. The original dataset comprises 2343 images, whereas we specifically utilized a subset of 50 images along with their corresponding masks for our experiments.

The FloodNet dataset comprises multiple classes or categories for semantic segmentation and object detection. These classes are represented in the table below (Table 4.1). The dataset provides valuable information for analyzing and understanding flooded areas, allowing for more accurate identification, segmentation, and detection of various objects or areas within the imagery—the assigned colors for each class aid in visually distinguishing them during the segmentation and object detection processes.

These classes represent different objects or areas within the flooded scenes and provide valuable information for accurately identifying and assessing the extent of flooding and damages in the affected areas [84].

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 4.1: Class Definitions - FLOODNET Dataset.

The images in the FloodNet dataset are either resized or cropped to dimensions that are multiples of 256. They are subsequently stored in a designated directory. Each image undergoes a process where patches sized 256x256x3 are extracted and preserved. After cutting the images into 256x256 patches, the total number of images is 8415. Then, you separated the images into two categories: those containing useful information (images with masks having more than 5% non-zero pixels) and those without useful information. The number of useful images is 5066, while the number of useless images is 3349.

For model training, approximately **90%** of the available images from the training dataset, which amounts to **4559** images with a resolution of **256x256** pixels, are used for training the models. The remaining **10%** is reserved for the validation process which amounts to **507** images.

The training process involves using a batch size of **16** and running **100 epochs**. The number of steps per epoch is determined by dividing the number of training images by the batch size ($4559/16 = 284,94$).

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. Adam Optimizer is used during the process of assembling models. As for loss function, it uses a specialized function, sm.losses.categorical-focal-jaccard-loss. The model's performance is assessed using two metrics: accuracy, which measures the general correctness of predictions, and the IoU, which determines the overlap between predicted truth areas and the ground in the fragmentation function.

To ensure easy access without re-training the model from scratch, we store the models in hdf5 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 and review their performance at any desired time. Here are some prominent features of the dataset are shown in figure 4.1

Figure 4.1: Some representative images from theFloodNet dataset.

4.2.2 Backbone

In the context of deep learning, the term "backbone" refers to 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 VGG, ResNet, and Inception. These CNN architectures are the foundation for feature extraction and representation learning in various computer vision tasks, including object detection and semantic segmentation.

The choice of backbone architecture plays a crucial role in determining the performance and efficiency of the overall deep learning model.

4.2.3 Evaluation and comparison

The obtained performance metrics such as accuracy and IoU are given in (Table 4.2). In training our models, we have fixed the Hyperparameters of training so we can put all the models in the same training environment and evaluate them.

	Accuracy	**IoU**	**Val_Accuracy**	**Val_IoU**
Unet+Resnet34	0.7471	0.5855	0.7377	0.5731
Unet+VGG19	0.7221	0.5477	0.7036	0.5279
DeeplabV3	0.9528	0.8944	0.6367	0.4931

Table 4.2: Results of the performance metrics for each model for semantic segmentation

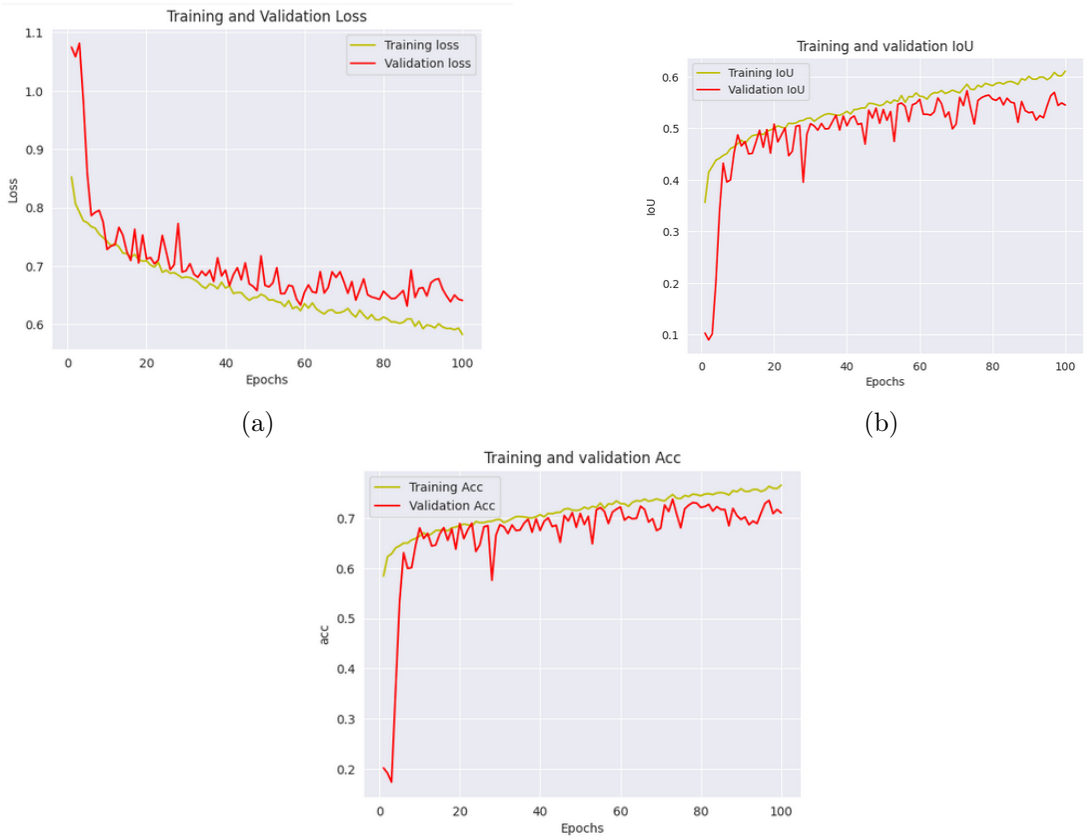
From Table 4.2, we can observe that the best results are achieved by the **Unet+Resnet34** model. This approach gives an IoU metric of **0.5855**, which is the best among all the studied models. The same trend can be noted with the evaluation using the accuracy metric, where the highest value is achieved by the **Unet+Resnet34** model with an accuracy value of **0.7471**.

Comparing the models, we can see that the **Unet+VGG19** model achieves slightly lower performance in terms of both accuracy and IoU metrics, with an accuracy value of **0.7221** and an IoU value of **0.5477**.

The **DeeplabV3** model is experiencing overfitting. Despite achieving relatively high accuracy of **0.9528** and a model's IoU of (**0.8944**), the obtained validation accuracy of **0.6367** and IoU of **0.4931** indicate that the Deeplabv3 model struggles with generalization.

Overall, the **Unet+Resnet34** model outperforms the other models in terms of both accuracy and IoU metrics, followed by the **Unet+VGG19** and **DeeplabV3** models, respectively.

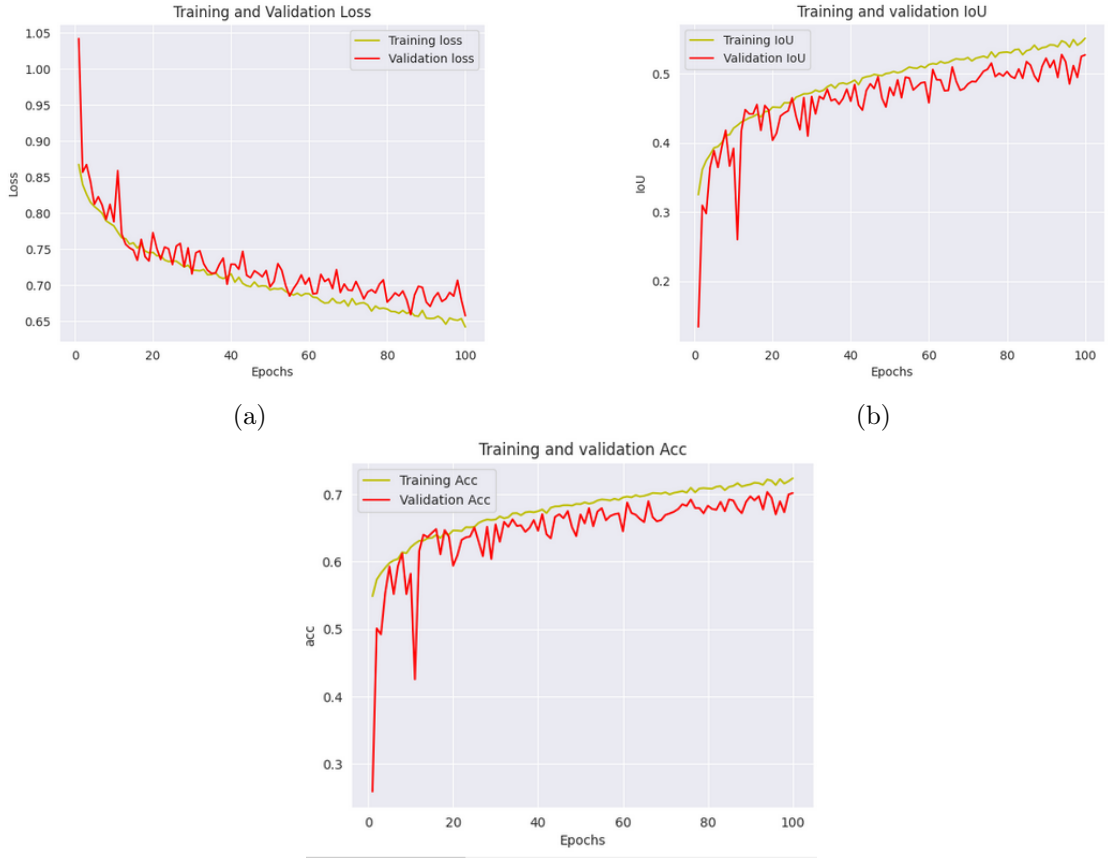
In figures 4.2, 4.3 and,4.4), we display the evolution of the values of the performance metrics in every epoch of the training and validation process in our models.



(a) (b)

(c)

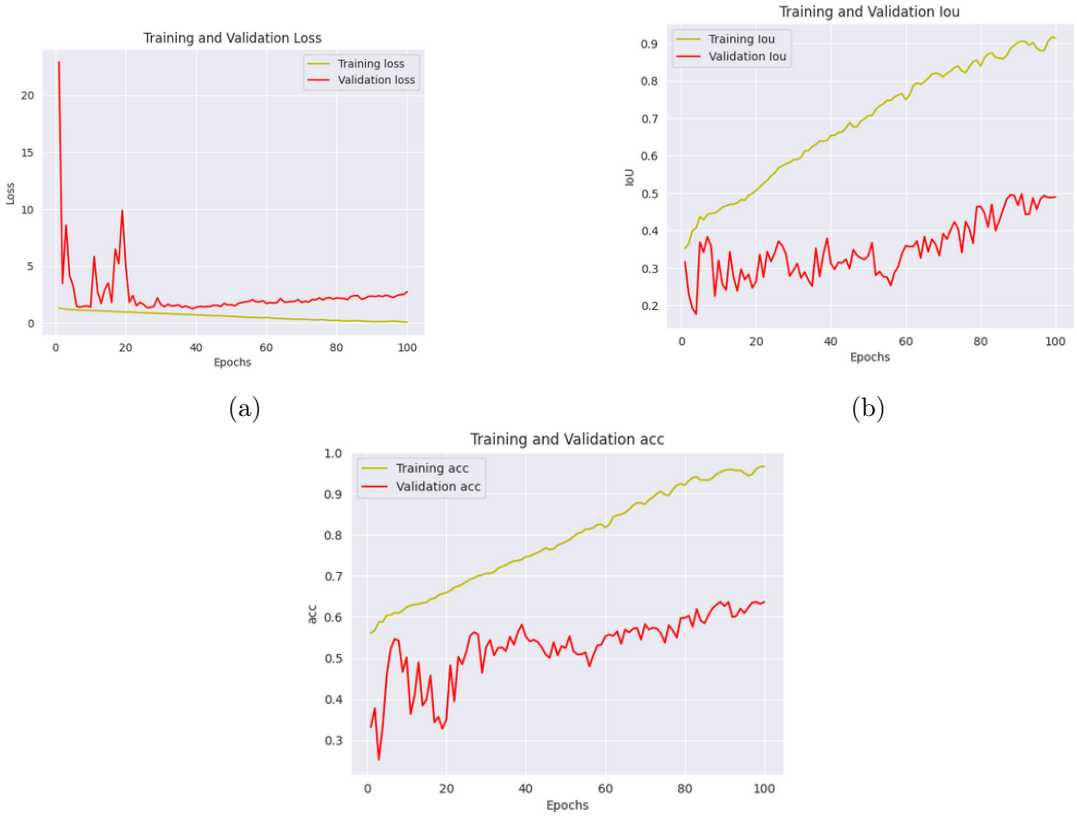
Figure 4.2: Evolution of the values of the performance metric in Unet+Resnet34



(a) (b)

(c)

Figure 4.3: Evolution of the values of the performance metric in Unet+VGG19

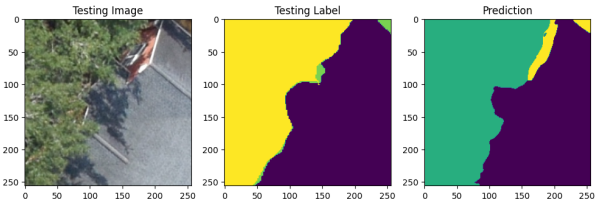


(a) (b)

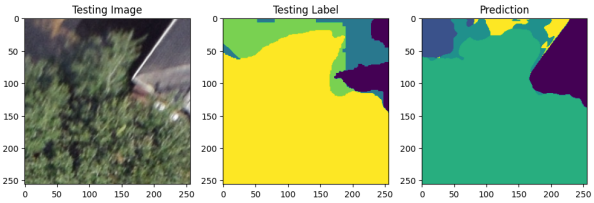
(c)

Figure 4.4: Evolution of the values of the performance metric in DeepLab.

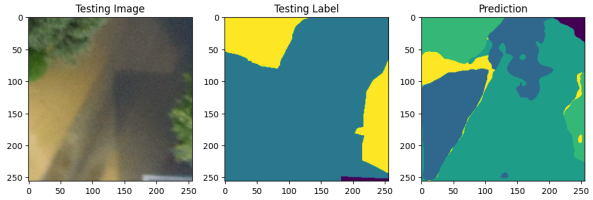
4.2.4 Visual results



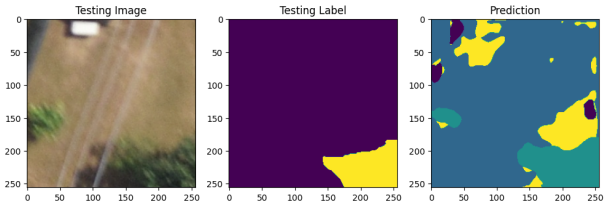
(a) Unet+Resnet34 prediction (Image1)



(b) Unet+Resnet34 prediction (Image2)

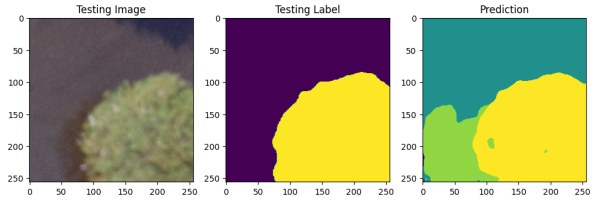


(c) Unet+Resnet34 prediction (Image3)

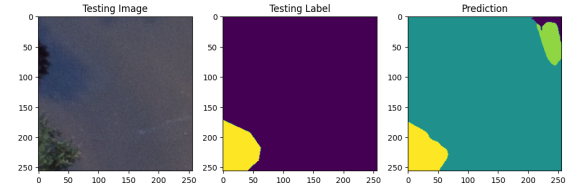


(d) Unet+Resnet34 prediction (Image4)

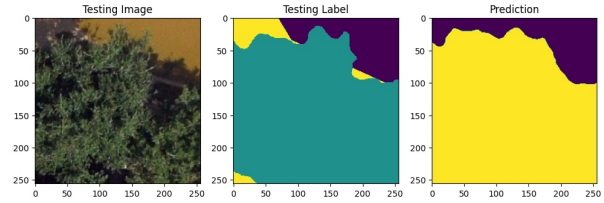
Figure 4.5: Visula results for the semantic segmentation using Unet+Resnet34 model.



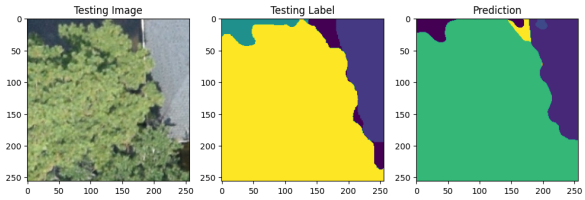
(a) Unet+VGG19 prediction (Image1)



(b) Unet+VGG19 prediction (Image2)



(c) Unet+VGG19 prediction (Image3)



(d) Unet+VGG19 prediction (Image4)

Figure 4.6: Visula results for the semantic segmentation using Unet+VGG19 model.

4.3 Experiments on the object detection

4.3.1 Dataset

4.3.1.1 RescueNet

"RescueNet" is a dataset comprising a diverse collection of images used for structural damage detection in building assessment. The dataset possesses several key characteristics that make it suitable for training object detection models and classifying building damages. Here are some prominent features of the dataset:

- Number of images: The dataset comprises 7016 distinct images representing various scenes of buildings.
- Data format: Images are annotated in the YOLOv8 format, a common format for machine learning tasks, especially object detection.
- Data augmentation: Data augmentation techniques were applied to each source image, creating multiple versions with various transformations. These transformations include a 50% probability of horizontal and vertical flips, equal

probability of one of three 90-degree rotations (none, clockwise, counter-clockwise), random rotation between -15 and +15 degrees, random shear horizontally and vertically between -15° and +15°, and random brightness adjustments between -25% and +25%.

- Data split: The dataset is divided into three main subsets: a training set containing 94% of the data, a validation set comprising 4% of the data, and a smaller test set accounting for 2% of the data.
- Damage classification: The dataset focuses on classifying damages in buildings into three primary classes: "Moderate-Damage," "No-Damage," and "Total-Destruction."
- Image resolution: All images in the dataset have a size of 640x640 pixels, making them suitable for object detection and damage classification tasks.

This dataset was created with the goal of training object detection models specifically for the task of identifying and classifying damages in buildings. It can be valuable for various applications, such as post-natural disaster damage assessment or building condition evaluation. Additionally, the dataset was created using the Roboflow platform, which facilitated the process of data collection, enhancement, and transformation. Figure 4.7 shows some representative images in the dataset.

Figure 4.7: Some representative images from the RescueNet dataset.

4.3.2 Evaluation

In the table 4.3, RescueNet data represent the analysis of damage to buildings based on the following three categories: "Moderate-Damage," "No-Damage," and "Total-Destruction". It shows the total number of instances in each category, as well as the probabilities for the selected bounding boxes, and the Recall values indicating the model's ability to detect true positives. It also includes the model's average precision in detecting the categories (MAP50) and in the range of IoU=0.50 to 0.95 (MAP50-95). High values in MAP and MAP50-95 indicate better performance in detecting the categories, while other values, such as R, indicate the model's effectiveness in identifying true positive cases.

Here are the detailed results:

Class	Instances	Box(p)	R	MAP50	MAP50-95
All	927	0.703	0.58	0.665	0.419
Moderate Damage	249	0.61	0.622	0.604	0.381
No Damage	299	0.756	0.559	0.806	0.576
Total Destruction	279	0.745	0.459	0.584	0.301

Table 4.3: Damage detection results using 8 batches and 25 epochs.

Figure 4.8 shows the information related to the manual labeling of the objects in this dataset. We present the subfigures in Figure 4.8 from left to right and top to bottom. The first subfigure shows the number of objects of each type in the dataset and indicates that the objects are dominated by "No-Damage". The second subfigure shows the size of the object bounding boxes in the dataset, and the coordinates of the centers of all object boxes are fixed at one point. The size of the object bounding box size shows that the dataset contains a large number of small-area objects. The third subfigure represents the distribution of the coordinates of the central points of the object identification funds, and it can be seen that the central points of the objects are scattered in the image data area. The fourth subfigure displays a scattered representation of the width and height corresponding to the object bounding boxes, with the darkest color at the bottom left corner of the plot. It further illustrates that the current dataset is predominantly composed of small objects. This reflects that the data was captured from a high vantage point using drones, allowing for the detection of objects despite their small size.

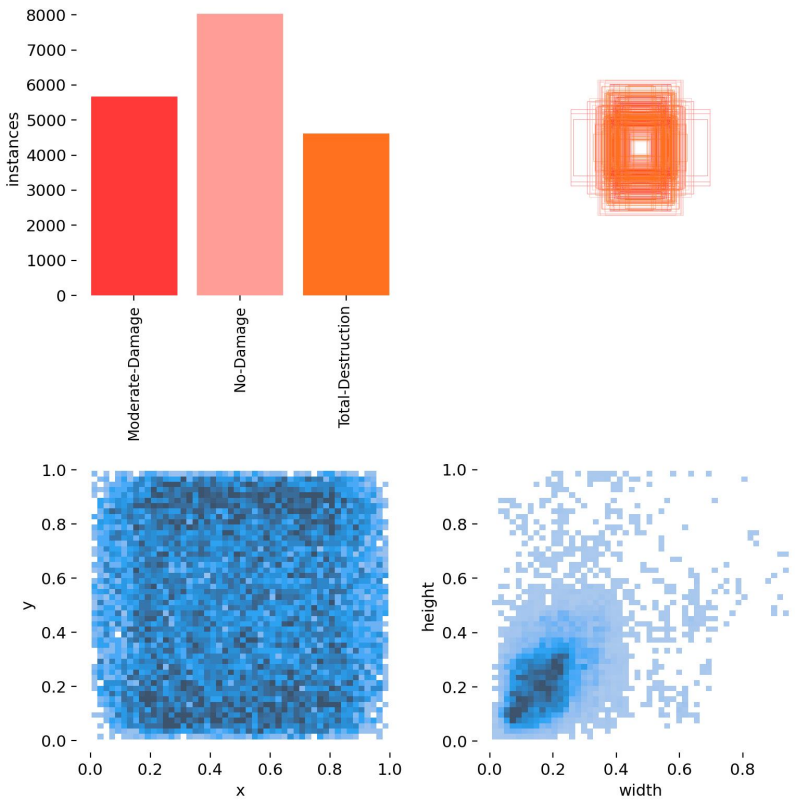


Figure 4.8: Information about the manual labeling of objects in RescueNet dataset.

Below in Figure 4.9 are the curves illustrating the model's performance in object detection during training and validation.

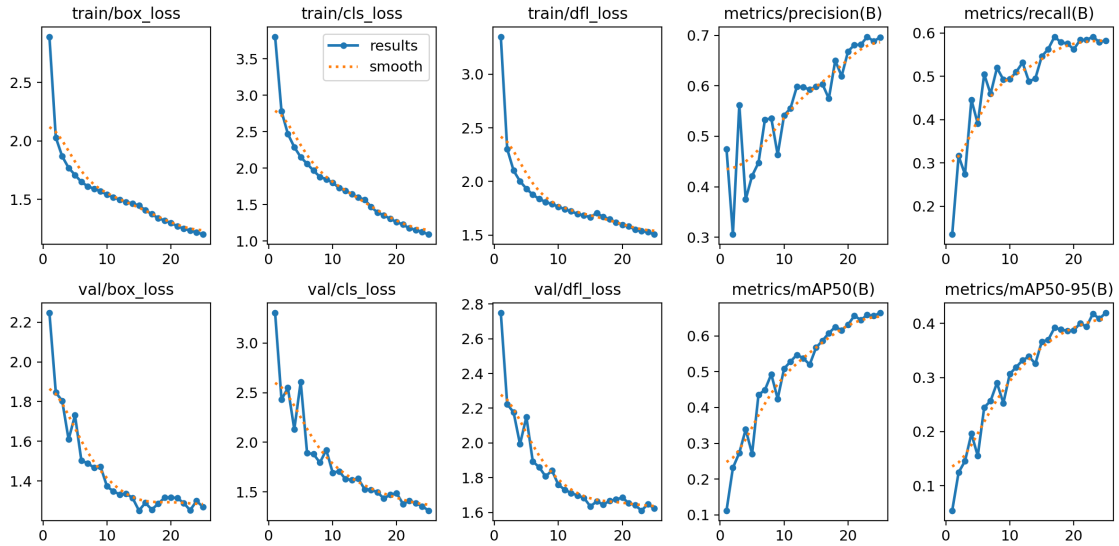


Figure 4.9: Graphs Performance Metrics Curves for Object Detection Model

Train box-loss, train cls-loss, train dfl-loss: These curves represent the model's loss during training. These curves gradually decrease, meaning the model is learning well and improving in predicting bounding boxes and classes.

Val box-loss, val cls-loss, val dfl-loss: These curves represent the model's loss on the validation dataset. These curves decrease gradually, meaning the model performs well on data it has not seen during training.

Metrics/precision(b), metrics/recall(b), metrics/map50(b), metrics/map50-95(b): These plots show how the model performs in precision, recall, and the average precision metric (MAP) for bounding boxes when using a specific IoU (for MAP50 and MAP50-95). These curves are increasing, so the model better predicts object locations.

4.3.3 Visual results

Figure 4.10 presents some visual results obtained by applying the YOLOv8 model to the RescueNet dataset.

Figure 4.10: Examples of object detection results.

4.4 Hardware used in deep learning

Deep learning algorithms require significant computational power to process large amounts of data and perform complex calculations. Various hardware components are commonly used in deep learning systems to accelerate the training and inference processes. This section will discuss three key hardware components: Central Processing Units (CPUs), Graphics Processing Units (GPUs), and Tensor Processing Units (TPUs).

4.4.1 Central processing units

A CPU is a computational device composed of multiple Arithmetic Logic Units (ALUs), a Control Unit, a Cache Memory, and a Dynamic Random-Access Memory (DRAM). Due to their superior power, CPUs enable computers to perform a wide range of tasks with precision and versatility. This versatility stems from the substantial memory capacity of CPUs, which can exceed 1TB of RAM, facilitating faster and lower latency retrieval of memory packages from RAM. In contrast, GPUs (Graphics Processing Units) do not currently meet the rigorous requirements of DL. Nonetheless, CPUs play a vital role in supporting GPUs by providing ample data and managing read/write operations between RAM/HDD during preparation stages [85].

4.4.2 Graphics processing units

In recent years, GPUs, which are primarily designed for generating polygon-based computer graphics, have witnessed significant advancements in computing power. This can be attributed to the increasing complexity and demand for realistic graphics in modern video games and graphic engines. Leading the market is NVIDIA, known for its GPUs equipped with thousands of cores optimized for high-performance computing. Interestingly, these processors have demonstrated their ability to handle neural network computations and matrix multiplications, extending their utility beyond graphics processing [86].

GPUs have become the standard choice for training deep learning systems like CNNs and RNNs. They offer high-speed processing, allowing for training on large batches of images in just milliseconds. However, GPUs have high power consumption, typically around 250 watts, requiring a full PC setup with an additional 150 watts. More advanced GPU systems can consume up to 400 watts. Popular GPUs for training DL neural networks include ZOTAC GeForce GTX 1070, ASUS ROG Strix Radeon RX 570, Gigabyte GeForce GT 710, and Sapphire Radeon Pulse RX 580.

4.4.3 Tensor processing units

Google has developed TPUs based on custom ASICs (Application-Specific Integrated Circuits) specifically designed to accelerate machine learning workloads. TPUs have been created with Google's extensive expertise and leadership in machine learning in mind. They excel at accelerating linear algebra computations, which are widely used in machine learning applications.

TPUs offer significant benefits when training large and complex neural network models. They greatly reduce the time required to achieve accurate results, enabling models that previously took weeks to train on other hardware platforms to converge in a matter of hours. TPUs in the cloud are optimized for individual workloads, ensuring efficient performance. In some cases, it may be appropriate to use GPUs or CPUs on Compute Engine instances for machine learning workloads. The choice of hardware depends on the specific requirements of the workload. It is important to select the appropriate hardware to achieve optimal performance [87].

4.5 Deep learning Software and tools

In this section, we will present the meanings and explanations of the programming languages, software, and tools that were employed in creating our application.

4.5.1 Python programming language

Python is a popular programming language known for its simplicity, readability, and ease of use, as well as its vast library of open-source packages and modules. It supports object-oriented and functional programming and provides built-in support for various data types, including lists, dictionaries, and tuples. Its dynamic type system allows for flexible and rapid development. Python is widely adopted by major tech companies, such as Google, Amazon, and Microsoft, and extensively used in academia, particularly in the fields of data science and machine learning [88].

Figure 4.11: Python logo

4.5.2 Google Colaboratory

Google Colaboratory, or Colab for short, is a free, cloud-based platform that allows users to write, execute, and share Python code. It provides a Jupyter Notebook environment, with the ability to access and run code on Google's cloud servers, as well as to collaborate with others in real time. Colab comes pre-installed with popular data science and machine learning libraries, including TensorFlow, Keras, and PyTorch, making it a popular choice among researchers and educators. Colab also allows users to upload and work with their own datasets and provides easy integration with Google Drive [89].

Figure 4.12: Google Colaboratory logo

4.5.3 Open Source Computer Vision Library

OpenCV is an open-source computer vision and machine learning software library. It provides a comprehensive set of tools for image and video processing, including various algorithms for image filtering, feature detection, object recognition, and

tracking. OpenCV is written in C++ but also has interfaces for other programming languages, including Python, Java, and MATLAB. The library is widely used in industry and academia, particularly in fields such as robotics, autonomous vehicles, and augmented reality [90],

Figure 4.13: OpenCV logo

4.5.4 TensorFlow

TensorFlow is a machine learning platform developed by Google that enables engineers and researchers to manipulate mathematical expressions over numerical tensors. It can automatically compute the gradient of any differentiable expression, making it highly suitable for machine learning. TensorFlow can run on CPUs, GPUs, and TPUs, and computation defined in TensorFlow can be easily distributed across many machines. TensorFlow applications can be exported to other runtimes, such as C++, JavaScript, or TensorFlow Lite, making it easy to deploy in practical settings. TensorFlow is more than a single library with a vast ecosystem of components developed by both Google and third parties. TensorFlow scales well and has been used to train the exaFLOPS extreme weather forecasting model on the IBM Summit supercomputer and to develop compute-intensive deep learning applications such as AlphaZero [91].

Figure 4.14: TensorFlow logo

4.5.5 Keras

Keras is a high-level deep learning API for Python, built on top of TensorFlow, that provides a convenient way to define and train deep learning models on top of different types of hardware (GPU, TPU, or CPU) and can be seamlessly scaled to thousands of machines. Keras prioritizes the developer experience and offers a wide range of different workflows, from high-level to low-level, enabling users to build and train models with varying levels of usability and flexibility [91].

Figure 4.15: Keras logo

4.5.6 Roboflow

Roboflow is a platform and toolset designed to streamline the process of preparing, annotating, and managing datasets for computer vision tasks. It offers various features, including data augmentation, annotation tools, dataset versioning, and integration with popular deep-learning frameworks. Roboflow aims to simplify and accelerate the development of machine learning models by providing a comprehensive data preparation and annotation solution.

Figure 4.16: Roboflow logo

4.5.7 Ultralytics

Ultralytics is a suite of tools and libraries in the field of artificial intelligence and machine learning. It aims to facilitate the development and training of deep neural network models, particularly in areas like image analysis and object detection. Ultralytics provides useful features and APIs that make it easy to conduct experiments and analyze results.

Figure 4.17: Ultralytics logo

Conclusion

This chapter shows the evaluation of the use of Unet+VGG, Unet+Resnet34, Deeplabv3 models in semantic segmentation as well as the application of YOLO model for object detection. The results indicated that Unet+Resnet34 yielded favorable results, while YOLOv8n showed acceptable performance in object detection.

General conclusion

In conclusion, this research focused on the effective assessment of the measurement of the proportion of damage to buildings and the assessment of damage after natural disasters using aerial image analysis. The main objective was to develop an automated system capable of detecting and classifying different types of damage, with particular emphasis on identifying affected areas. Significant progress has been made in the field of semantic segmentation and object detection through unmanned aerial vehicles (UAVs) and deep learning techniques, specifically synthetic neural networks (CNNs). Various models, including Unet+Resnet34, Unet+VGG, Deeplabv3, and YOLOv8n, were evaluated in this study. Each model has its strengths and limitations, and its performance has been assessed to achieve accurate and reliable results in detecting and classifying different types of damage.

In the case of training the model YOLOv8n, it is worth noting that we opted for a training duration of only 25 epochs due to hardware resource constraints and time limitations. While it would have been beneficial to extend this to 50 epochs for potentially improved results and a more refined model, practical considerations necessitated the abbreviated training period.

Research results demonstrate the possibility of using aerial image analysis and deep learning techniques to assess damage after natural disasters efficiently. The developed system provides valuable insights into the extent and types of damage, helping with rapid response and recovery efforts. However, further research and improvements are needed to enhance the system's performance and expand its application in real-world scenarios.

By leveraging the capabilities of drones and deep learning algorithms, this research contributes to disaster management and paves the way for more effective and automated assessment methods. The application of such technologies can greatly help mitigate the effects of natural disasters and facilitate timely responses and recovery. Despite resource and time constraints, this study represents an important step forward in seeking more efficient disaster relief efforts.

Bibliography

[1] Himadri Nath Saha et al. "Waste management using internet of things (iot)". In: *2017 8th annual industrial automation and electromechanical engineering conference (IEMECON)*. IEEE. 2017, pp. 359–363.

[2] Kathryn Hay and Katheryn Margaret Pascoe. "Engaging social workers in disaster management: Case studies from New Zealand". In: *International Journal of Disaster Risk Reduction* 74 (2022), p. 102941.

[3] Agoston Restas et al. "DrA1one applications for supporting disaster management". In: *World Journal of Engineering and Technology* 3.03 (2015), p. 316.

[4] Vinay Dubey, Prashant Kumar, and Naveen Chauhan. "Forest fire detection system using IoT and artificial neural network". In: *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2018, Volume 1*. Springer. 2019, pp. 323–337.

[5] Dai Quoc Tran et al. "Damage-map estimation using UAV images and deep learning algorithms for disaster management system". In: *Remote Sensing* 12.24 (2020), p. 4169.

[6] Jun Rentschler, Melda Salhab, and Bramka Arga Jafino. "Flood exposure and poverty in 188 countries". In: *Nature communications* 13.1 (2022), p. 3527.

[7] Agoston Restas. "Water related disaster management supported by drone applications". In: *World Journal of Engineering and Technology* 6 (2018), pp. 116–126.

[8] Jane Bullock, George Haddow, and Damon P Coppola. *Introduction to emergency management*. Butterworth-Heinemann, 2017.

[9] Naqqash Dilshad et al. "Applications and challenges in video surveillance via drone: A brief survey". In: *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE. 2020, pp. 728–732.

[10] Sharifah Mastura Syed Mohd Daud et al. "Applications of drone in disaster management: A scoping review". In: *Science & Justice* 62.1 (2022), pp. 30–42.

[11] Vishal Gewali and Sanjeeb Prasad Panday. "Deep Neural Networks for Wild Fire Detection and Monitoring with UAV". In: *Advanced Communication and Intelligent Systems: First International Conference, ICACIS 2022, Virtual Event, October 20-21, 2022, Revised Selected Papers*. Springer. 2023, pp. 411–423.

[12] Reza Shakeri et al. "Design challenges of multi-UAV systems in cyber-physical applications: A comprehensive survey and future directions". In: *IEEE Communications Surveys & Tutorials* 21.4 (2019), pp. 3340–3385.

[13] Faine Greenwood, Erica L Nelson, and P Gregg Greenough. "Flying into the hurricane: A case study of UAV use in damage assessment during the 2017 hurricanes in Texas and Florida". In: *PLoS one* 15.2 (2020), e0227808.

[14] Sven Mayer, Lars Lischke, and Paweł W Woźniak. "Drones for search and rescue". In: *1st International Workshop on Human-Drone Interaction*. 2019.

[15] Jose Anand et al. "Drones for Disaster Response and Management". In: *Internet of Drones*. CRC Press, pp. 177–200.

[16] Qian Wang et al. "An Overview of Emergency Communication Networks". In: *Remote Sensing* 15.6 (2023), p. 1595.

[17] Milan Erdelj et al. "Help from the sky: Leveraging UAVs for disaster management". In: *IEEE Pervasive Computing* 16.1 (2017), pp. 24–32.

[18] Aakash Sehrawat, T Anupriya Choudhury, and Gaurav Raj. "Surveillance drone for disaster management and military security". In: *2017 international conference on computing, communication and automation (ICCCA)*. IEEE. 2017, pp. 470–475.

[19] Kimon P Valavanis and George J Vachtsevanos. *Handbook of unmanned aerial vehicles*. Vol. 1. Springer, 2015.

[20] Vandana Mohindru et al. *Unmanned Aerial Vehicles for Internet of Things (IoT): Concepts, Techniques, and Applications*. John Wiley & Sons, 2021.

[21] Syed Agha Hassnain Mohsan et al. "Towards the unmanned aerial vehicles (UAVs): A comprehensive review". In: *Drones* 6.6 (2022), p. 147.

[22] Cinzia Albertini et al. "Detection of Surface Water and Floods with Multispectral Satellites". In: *Remote Sensing* 14.23 (2022), p. 6005.

[23] Huang Yao, Rongjun Qin, and Xiaoyu Chen. "Unmanned aerial vehicle for remote sensing applications—A review". In: *Remote Sensing* 11.12 (2019), p. 1443.

[24] Shervin Minaee et al. "Image segmentation using deep learning: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* (2021).

[25] Jaskirat Kaur and Williamjeet Singh. "Tools, techniques, datasets and application areas for object detection in an image: a review". In: *Multimedia Tools and Applications* 81.27 (2022), pp. 38297–38351.

[26] Zhong-Qiu Zhao et al. "Object detection with deep learning: A review". In: *IEEE transactions on neural networks and learning systems* 30.11 (2019), pp. 3212–3232.

[27] W Yan. *Computational methods for deep learning*. Springer, 2021.

[28] Iqbal H Sarker. "Deep learning: a comprehensive overview on techniques, taxonomy, applications, and research directions". In: *SN Computer Science* 2.6 (2021), p. 420.

[29] Nitin Kumar Chauhan and Krishna Singh. "A review on conventional machine learning vs deep learning". In: *2018 International conference on computing, power and communication technologies (GUCON)*. IEEE. 2018, pp. 347–352.

[30] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[31] Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning Publications, 2020.

[32] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks". In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[33] P Sibi, S Allwyn Jones, and P Siddarth. "Analysis of different activation functions using back propagation neural networks". In: *Journal of theoretical and applied information technology* 47.3 (2013), pp. 1264–1268.

[34] Yuhan Bai. "RELU-Function and Derived Function Review". In: *SHS Web of Conferences*. Vol. 144. EDP Sciences. 2022, p. 02006.

[35] Nils Bjorck et al. "Understanding batch normalization". In: *Advances in neural information processing systems* 31 (2018).

[36] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[37] J Huber. "Batch normalization in 3 levels of understanding". In: *Towards Data Science* 6 (2020).

[38] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[39] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[40] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[41] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2)". In: *Doklady an ussr*. Vol. 269. 1983, pp. 543–547.

[42] Devansh- Machine Learning Made Simple. *How does batch size impact your model learning*. `https://medium.com/geekculture/how-does-batch-size-impact-your-model-learning-2dd34d9fb1fa`. Dec. 2022.

[43] Andrew Ferlitsch. *Deep Learning Patterns and Practices*. " Manning.", 2021, pp. 91–95.

[44] Halit Apaydin et al. "Comparative analysis of recurrent neural network architectures for reservoir inflow forecasting". In: *Water* 12.5 (2020), p. 1500.

[45] Mamta Mittal, Rajiv Ratn Shah, and Sudipta Roy. *Cognitive Computing for Human-Robot Interaction: Principles and Practices*. Academic Press, 2021.

[46] 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.

[47] Alhassan Mumuni and Fuseini Mumuni. "Data augmentation: A comprehensive survey of modern approaches". In: *Array* (2022), p. 100258.

[48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[49] Shervin Minaee et al. *Image Segmentation Using Deep Learning: A Survey*. 2020. arXiv: 2001.05566 [cs.CV].

[50] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation. arXiv 2017". In: *arXiv preprint arXiv:1704.06857* (2020).

[51] Raimi Karim. "Illustrated: 10 CNN architectures. towardsdatascience. com". In: *2019, https://towardsdatascience. com/illustrated-10-cnnarchitectures95d78ace614d* (2019).

[52] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[53] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[54] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[55] Sadegh Pasban et al. "Infant brain segmentation based on a combination of VGG-16 and U-Net deep neural networks". In: *IET Image Processing* 14.17 (2020), pp. 4756–4765.

[56] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[57] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[58] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[59] Sik-Ho Tsang. "Review: MobileNetV1–depthwise separable convolution (lightweight model)". In: *Towards Data Science* (2018).

[60] 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. 3904–3913.

[61] Ilias Papadeas et al. "Real-time semantic image segmentation with deep learning for autonomous driving: A survey". In: *Applied Sciences* 11.19 (2021), p. 8802.

[62] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[63] 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.

[64] Ozan Oktay et al. "Attention u-net: Learning where to look for the pancreas". In: *arXiv preprint arXiv:1804.03999* (2018).

[65] Liang-Chieh Chen et al. “Semantic image segmentation with deep convolutional nets and fully connected crfs”. In: *arXiv preprint arXiv:1412.7062* (2014).

[66] Liang-Chieh Chen et al. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *IEEE Transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[67] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

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

[69] François Chollet. “Xception: Deep learning with depthwise separable convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1251–1258.

[70] 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.

[71] Steven A Hicks et al. “On evaluation metrics for medical applications of artificial intelligence”. In: *Scientific Reports* 12.1 (2022), p. 5979.

[72] Ran Shi, King Ngi Ngan, and Songnan Li. “Jaccard index compensation for object segmentation evaluation”. In: *2014 IEEE international conference on image processing (ICIP)*. IEEE. 2014, pp. 4457–4461.

[73] Yassine Bouafia and Larbi Guezouli. “An overview of deep learning-based object detection methods”. In: (2018).

[74] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[75] Shubham Shinde, Ashwin Kothari, and Vikram Gupta. “YOLO based human action recognition and localization”. In: *Procedia computer science* 133 (2018), pp. 831–838.

[76] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.

[77] Joseph Redmon and Ali Farhadi. “Yolov3: An incremental improvement”. In: *arXiv preprint arXiv:1804.02767* (2018).

[78] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv preprint arXiv:2004.10934* (2020).

[79] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[80] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[81] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[82] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[83] Jun Deng et al. “A review of research on object detection based on deep learning”. In: *Journal of Physics: Conference Series*. Vol. 1684. 1. IOP Publishing. 2020, p. 012028.

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

[85] Yasmeen Khaled and Amr Kayid. “Performance of CPUs/GPUs for Deep Learning workloads”. In: *Artificial General Intelligence*. May 2018. DOI: 10.13140/RG.2.2.22603.54563.

[86] Eugenio Culurciello. *Hardware for Deep Learning*. Towards Data Science. Mar. 2017. URL: https://towardsdatascience.com/hardware-for-deep-learning-8d9b03df41a.

[87] Google Cloud. *Cloud Tensor Processing Units (TPUs)*. May 2016. URL: https://cloud.google.com/tpu/docs/tpus.

[88] Python Software Foundation. *Python. What is Python? Executive Summary*. https://www.python.org/doc/essays/blurb/. accessed 2023-05-10.

[89] *Google Colaboratory*. Website. Accessed: May 10, 2023. URL: https://colab.research.google.com/.

[90] Gary Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* 25.11 (2000), pp. 120–123.

[91] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.