POLITECNICO DI TORINO

Master's Degree in MECHATRONIC ENGINEERING

Master's Degree Thesis

Industrial Facilities Rooftop Detection Using Aerial Imagery

Supervisors

Prof. MARCELLO CHIABERGE

Dott. FRANCESO MESSINA

Dott. UMBERTO ALBERTIN

Candidate

ALI ABBASS

DECEMBER 2024

Abstract

Inspection is an essential phase for identifying potential defects and minimizing risks, especially in critical building components. The primary focus of this work is the monitoring of industrial facility rooftops. Rooftops are the main part of any building or facility, and roof defects pose risks to personnel and equipment, therefore regular inspections enable timely repairs or replacements, preserving the proper functionality and safety of industrial facilities. In the past, these inspections were carried out through manual surveys, which were time-consuming, prone to human error, costly, and often hazardous for workers.

The advent of new technologies, such as Unmanned Aerial Systems (UAS), has proven to be an effective option for overcoming these limitations. Unmanned Aerial Vehicles (UAVs), commonly known as drones, provide potential as effective tools for infrastructure inspections. UAVs enable visual inspections of structures while limiting the need for direct human involvement, reducing inspection costs and time and minimizing danger to workers. UAVs also provide access to unique views of urban imagery that have traditionally been difficult to capture. For instance, rooftops of industrial facilities often contain equipment such as pipelines, solar panels, water or oil tanks, and heating, ventilation, and air conditioning systems (HVAC). Additionally, some areas may be difficult to access due to steep slopes or blocking obstacles.

In recent years, significant developments in deep learning, particularly in Convolutional Neural Network (CNN) architectures and related software and hardware, have helped in the analysis of large datasets, achieving outstanding results in many complex tasks. The methodology of combining UAVs with deep learning offers an effective solution for monitoring rooftops in industrial facilities, especially for complex structures.

In this work, we use the potential of this methodology by proposing four state-of-the-art CNN architectures, which are Mask R-CNN, DeepLabV3+, U-Net, and YOLOv8, to process aerial images captured by a UAV for automated segmentation and detection of flat and sloped roofs in a synthetic facility map.

Table of Contents

List of Tables

List of Figures

Acronyms

AI
artificial intelligence

NN
neural network

CNN
convolutional neural network

DL
deep learning

ML
machine learning

CV
computer vision

RGB
red, green, and blue

ANN
artificial neural network

UAV
unmanned aerial vehicle

mIoU
mean intersection over union

FC

fully connected

SGD

stochastic gradient descent

Chapter 1

Introduction

The accurate identification and classification of rooftops on industrial facilities are of increasing interest for various applications, including urban planning, facility maintenance, environmental monitoring, and solar panel installation. Industrial rooftops often cover large areas, making them ideal locations for solar panel installations, green roofs, and rainwater harvesting systems. Identifying rooftop types and accurately defining their boundaries is essential for optimizing these uses. Additionally, effective rooftop detection is crucial for facility maintenance, as it assists in assessing rooftop condition, identifying damage, and scheduling repairs. Traditional rooftop detection methods, often relied on satellite imagery or on-site inspections, were limited by resolution constraints and posed health and safety risks, especially when inspectors had to work at heights or in hard-to-reach areas or for long hours to accomplish precise checks, which highlighted the need for more efficient and automated approaches. These challenges have led to a growing interest in using new technologies to enhance safety, speed, and accuracy for this type of application.

Unmanned Aerial Vehicles (UAVs), have emerged as a transformative technology for data acquisition in industrial and urban environments, enabling the collection of high-resolution imagery at relatively low cost and with minimal disruption to facility operations. UAVs provided flexibility in capturing images from different heights and angles, making them ideal for industrial sites where rooftop visibility may vary due to structural complexities or environmental factors.

Recent advances in deep learning, particularly Convolutional Neural Networks (CNNs), combined with improvements in high-performance computing, especially through Graphics Processing Units (GPUs), have significantly expanded automated detection capabilities. CNNs possess the ability to learn complex features from input images and to process large amounts of data effectively. While various algorithms have been introduced in the literature for detecting and classifying buildings, facades, rooftops, relatively few studies focused specifically on industrial

facilities rooftop detection. CNN architectures consistently outperformed traditional methods in both instance and semantic segmentation, as well as object detection.

This work aimed to develop a methodology for detecting and classifying industrial facility rooftops using deep learning. Four state-of-the-art networks: DeepLabv3+, UNet, Mask R-CNN, and YOLOv8, were trained on a dataset comprising both synthetic and real images. Each model was evaluated based on its ability to detect two rooftop types, flat and gable. By comparing model performance across tasks, this research sought to identify the strengths and limitations of each architecture within the context of aerial industrial imagery. The findings of this work may extend beyond industrial facilities, as the proposed methodology could be adapted for similar applications in other urban or rural areas.

This thesis is organized as follows: Chapter 2 presents the literature review of UAVs, computer vision, deep learning, and CNNs across a wide range of applications, beyond just industrial facilities or rooftop detection. Chapter 3 details the methodology, covering data collection and preparation, as well as model training and evaluation. Chapter 4 discusses the results of training and evaluates the performance of the four CNN models.

Chapter 2

Materials and Methods

2.1 State-of-the-Art

Inspection, monitoring, and maintenance are essential aspects of managing large-scale industrial facilities, urban infrastructure, and critical assets. These activities ensure the operational efficiency, safety, and longevity of systems ranging from power plants and factories to transportation networks, and utilities. In industries where equipment and infrastructure are subject to constant wear, early detection of issues such as structural damage, system failures, or inefficiencies can significantly reduce downtime, prevent costly repairs, and ensure safety. Traditional inspection methods, such as manual checks or stationary monitoring, often face limitations and challenges, especially when inspecting complex or hard-to-reach areas. To address these challenges, advancements in technology, particularly in unmanned aerial vehicles (UAVs) and automated data analysis, have revolutionized the way inspections are conducted across various industries.

UAVs have emerged as a transformative tool in the field of industrial inspections, offering unparalleled flexibility, cost-effectiveness, and safety. UAVs can access locations that are difficult or hazardous for human inspectors to reach, such as tall buildings, bridges, powerlines, and remote infrastructure. By providing real-time, high-resolution images and data from various sensors, UAVs enable inspectors to assess infrastructure conditions with high precision. The ability to carry a variety of sensors, including thermal, LiDAR, multispectral, and RGB cameras, allows UAVs to capture a wide range of data about the infrastructure they are monitoring. This capability is invaluable not only in structural inspections but also in areas such as environmental monitoring, disaster management, and agriculture.

The integration of deep learning technologies, particularly Convolutional Neural Networks (CNNs), has significantly enhanced the potential of UAVs. CNNs, known for their ability to analyze large amounts of image data, have been instrumental

in automating the detection and classification of various features captured by UAVs. These models excel at identifying patterns and anomalies within images, enabling precise categorization and analysis. CNN-based approaches are widely used in various fields, from detecting cracks and corrosion in industrial structures to identifying vegetation stress in agricultural areas. One of the most notable applications is the analysis of aerial imagery to monitor and assess the condition of roofs, facades, and other critical elements of infrastructure.

When focusing on industrial facilities, the roof is a particularly important component that requires regular inspection due to its exposure to harsh environmental conditions. Over time, roofs may develop issues such as leaks, cracks, or structural weakness, which can affect the integrity of the entire building. The complexity of industrial rooftops, with their varying shapes, sizes, and materials, presents challenges for traditional inspection methods. Inspecting such areas manually can be time-consuming, dangerous, and costly, especially for large or hard-to-reach facilities. UAVs, with their flexibility and ability to capture high-resolution images from multiple angles, are well-suited for rooftop inspections, allowing for a thorough assessment without the need for personnel to access the site physically.

By combining UAVs with advanced image processing techniques, the detection of structural issues, such as damage or wear, becomes more efficient and accurate. Deep learning models, such as Mask R-CNN, YOLOv8, U-Net and DeepLabv3+, are increasingly being employed to analyze aerial imagery for automatic detection and classification of rooftop features. Semantic segmentation models, such as DeepLabv3+ and U-Net, are widely adopted for their ability to perform pixel-level classification, allowing for precise delineation of rooftop boundaries and identification of features. Mask R-CNN extends the capabilities of traditional object detection models by incorporating pixel-level segmentation for more detailed instance detection. YOLOv8, a modern object detection model, offers fast and efficient detection capabilities. Each of these models has been proven effective in various computer vision tasks and has been adapted for use in aerial image analysis, such as detecting and classifying rooftops. Automated analysis accelerates the inspection process, minimizes human error, and improves assessment accuracy. By quickly identifying potential problems, UAVs can help facility managers make informed decisions about maintenance priorities and resource allocation.

However, the adoption of UAVs and deep learning in industrial inspections is not without its challenges. Variability in infrastructure, such as differences in rooftop designs, materials, and environmental conditions, poses challenges in creating models that generalize effectively across diverse settings. Furthermore, factors such as lighting conditions, weather, and seasonal changes can affect the quality of the imagery captured, complicating the detection and analysis of subtle defects. The requirement for extensive annotated datasets for training deep learning models is another challenge, as labeling vast volumes of aerial imagery can be labor-intensive

and costly.

In conclusion, the combination of UAV technology and deep learning has significantly enhanced the capability of inspections across various sectors. While these advancements have already demonstrated their effectiveness across various domains, the potential for enhancing maintenance and monitoring practices remains significant. As challenges related to data variability and model generalization are addressed, UAVs equipped with deep learning models will continue to play a pivotal role in transforming industrial inspections, particularly for critical infrastructure like rooftops. The increased efficiency, safety, and cost-effectiveness provided by these technologies are poised to make inspections more accurate and accessible, ultimately improving the management and sustainability of industrial facilities worldwide.

2.2 Literature Review

Human vision extends beyond the mere function of our eyes, it encompasses our abstract understanding of concepts and personal experiences gained through countless interactions with the world. Historically, computers could not think independently. However, recent advancements have given rise to computer vision, a technology that mimics human vision to enable computers to perceive and process information similarly to humans. Computer vision has witnessed remarkable advancements fueled by breakthroughs in artificial intelligence and computing capabilities [1].

The history of computer vision dates back to the 1950s, when early experiments involved simple pattern recognition. It significantly advanced in the 1970s with the development of the first algorithms capable of interpreting typed and handwritten text. The introduction of the first commercial machine vision systems in the early 1980s marked another key milestone, primarily used in industrial applications for inspecting products. The field saw rapid growth with the advent of more powerful computers and the development of more complex algorithms in the 1990s and 2000s. The real breakthrough came with deep learning in the 2010s, particularly with the use of Convolutional Neural Networks (CNNs), which dramatically improved the accuracy and capabilities of computer vision systems, expanding their application to almost every industry today [1].

One compelling application of computer vision lies in its integration with unmanned aerial vehicles (UAVs), particularly for inspecting challenging environments. A notable study by Liu et al. [2] demonstrated the use of UAVs for curtain wall inspections at construction sites, establishing their effectiveness for inspecting commercial building sites. Similarly, Winkvist et al. [3] presented an autonomous UAV design for inspection of indoor locations that are GPS denied and have many

unchartered obstacles, without communication from a ground station. Silveira et al. [4] compared UAV-based and traditional building inspections, emphasizing benefits such as improved accessibility and reduced safety risks. The maneuverability of UAVs makes them ideal for capturing images of inaccessible features, as highlighted in research conducted by Morgenthal and Hallermann [5]. They discussed the possible applications of utilising UAVs for capturing detailed images for condition analysis of structures, reducing the time and cost of inspections. However, these benefits came with challenges, as UAVs were susceptible to environmental conditions such as wind velocity. Zhuo et al. [6] proposed a building segmentation algorithm based on building boundary constraints using oblique UAV images and optimized building footprint generation from OpenStreetMap (OSM). Bown and Miller [7] conducted an in-depth investigation into the use of UAVs for roof inspection and detection, covering the historical context of roof inspections, advancements in UAV technology, and criteria for selecting suitable UAVs for such tasks. The authors described several types of UAVs, detailing their costs, battery life, and potential operational issues. They also stressed the critical criteria required for achieving the best image quality during inspection.

The effectiveness of UAV-based inspection methods has also been extensively analyzed along with deep learning. Spasov et al. [8] proposed an optimized solution for the classification of rooftops from aerial imagery based on a deep learning model using CNNs. The dataset consisted of 3,517 rooftop images. The network used a fine-tuned ResNet-101 and its performance was compared to five other state-of-the-art models: Xception65, EfficientNetB7, Visual Transformer (ViT), Bidirectional Encoder representation from Image Transformers (BEiTv2) and hybrid ViT with ResNet-50. The proposed model showed the highest performance achieving an accuracy of 84%, a weighted F1-score of 85.2%, and a recall of 82%.

Deep learning has further transformed computer vision by addressing tasks like image classification, object detection, and semantic segmentation. Guo et al. (2016) [9] compared the literature and their respective performance on different CV tasks, including image classification, object detection, image retrieval, semantic segmentation, and human pose estimation. By comparing CNN, RBM, Autoencoder, and Sparse Coding, they finally concluded that CNN was the most suitable architecture for CV. Due to the limitation of precisions and model sizes at that time, there were several challenges in practical application. Chai and Li (2019) [10] and Huang et al. (2020) [11] highlighted the penetration of deep learning into fields such as natural language processing (NLP), finance, and banking. Nurkarim and Wijayanto (2023) [12] investigated the potential of deep learning techniques and high spatial resolution remote sensing datasets to automate the detection and computation of building coverage areas. The integration of UAV technology and deep learning has paved the way for groundbreaking solutions in roof detection and segmentation. CNNs have been effectively employed to classify, detect, and

analyze roof structures, ranging from rural villages (Boateng et al. [13]) to dense urban centers, CNNs have been employed to classify, detect, and analyze roof structures effectively. Research by Kim et al. [14] has further validated CNNs for roof material classification, achieving accuracy improvements of 5–7% over earlier methods. These advancements demonstrate the transformative potential of combining UAVs and deep learning in automating and optimizing inspection processes across diverse environments.

These advancements in computer vision, particularly in deep learning, have been driven largely by the development and optimization of Convolutional Neural Networks (CNNs). Renowned for their ability to automatically learn spatial hierarchies of features, CNNs have revolutionized numerous domains due to their high performance and adaptability. CNNs have found extensive application across a wide range of fields, demonstrating their versatility and efficacy. In computer vision, they have powered tasks like image classification [15, 16, 17], object detection [18], and face recognition [19, 20]. Beyond visual data, CNNs have also been applied to speech recognition [21], traffic sign recognition [22], and even medical diagnostics, such as identifying diabetic retinopathy [23] or recognizing facial expressions [24].

Convolutional Neural Network (CNN) is a well-known deep learning architecture inspired by the natural visual perception mechanism of living creatures. In 1959, Hubel & Wiesel [25] found that cells in animal visual cortex were responsible for detecting light in receptive fields. Inspired by this discovery, Kunihiko Fukushima proposed the neocognitron (a hierarchical, multilayered artificial neural network) in 1980 [26], which could be regarded as the predecessor of CNN. In 1990, LeCun et al. [27] published the seminal paper establishing the modern framework of CNN, and later improved it in [28]. They developed a multi-layer artificial neural network called LeNet-5 which could classify handwritten digits. Like other neural networks, LeNet-5 had multiple layers and could be trained with the backpropagation algorithm [29]. It was able to obtain effective representations of the original image, making it possible to recognize visual patterns directly from raw pixels with little-to-none preprocessing. A parallel study of Zhang et al. [30] used a shift-invariant artificial neural network (SIANN) to recognize characters from an image. However, due to the lack of large training data and computing power at that time, their networks could not perform well on more complex problems, e.g., large-scale image and video classification. Since 2006, many methods have been developed to overcome the difficulties encountered in training deep CNNs [31]. Most notably, Krizhevsky et al. proposed a classic CNN architecture and showed significant improvements upon previous methods on the image classification task. The overall architecture of their method, AlexNet, [32], is similar to LeNet-5 but with a deeper structure. With the success of AlexNet, many works have been proposed to improve its performance. From the evolution of the architectures, a typical trend was that the networks have been getting deeper, for example, ResNet,

which won the champion of ILSVRC 2015, was about 20 times deeper than AlexNet and 8 times deeper than VGGNet. By increasing depth, the network could better approximate the target function with increased nonlinearity and achieve more robust feature representations. However, it also increased the complexity of the network, making it more difficult to optimize and more prone to overfitting [33].

In industrial applications, they have been employed to classify faults in semiconductor manufacturing processes [34], highlighting their impact on improving precision in critical domains. The strength of CNNs lies in their ability to automatically learn and extract hierarchical features, which has been further enhanced by advancements in architecture. For example, Krizhevsky et al. [35] optimized CNN training through non-saturating neurons and dropout regularization to reduce overfitting, achieving remarkable performance in large-scale image classification.

In roof and building detection, CNN models have consistently demonstrated high accuracy and reliability in extracting structural features. Gao et al. (2021) [36] approached a building rooftop detection as an instance segmentation problem and proposed a region-based deep learning approach to extract building rooftop based on the Mask Regional Convolutional Neural Network (Mask R-CNN) framework. This study yielded strong results with a precision, recall, and F1-score of 92%, 86.6%, and 89.1%, respectively. Cai et al. (2020) [37] investigated deep neural networks for rapid and accurate detection of building rooftops in aerial orthoimages. They trained the data using three deep learning models, U-Net, DeepLabv3+ and Fully Convolutional Network (FCN). In the first training, the best result was achieved by DeepLabv3+ with 63.8% in IoU, 77.8% in mean IoU, 74% in precision, and 78% in F1-score. In the second training, after improving the performance with focal loss, training loss was greatly cut down and the convergence rate experienced a significant growth which resulted in a better rooftop detection and a higher performance, the best result was also achieved by DeepLabv3+ with 65.4% in IoU, 79% in mIoU, 74% in precision, and 79.1% in F1-score. Castello et al. (2019) [38] executed a supervised CNN-based method for pixel-wise segmentation of rooftop solar panels and size estimation. They relied on aerial photos to create their dataset. The study showed that this model was able to automatically detect the solar panels with an accuracy of 0.94 and an IoU of 0.64. Other efforts, such as those by Hang and Cai [39], demonstrated how tailored CNNs could identify industrial and residential roofs, the assessment results proved the effectiveness of the proposed method, with approximately 91% and 88% quality rates in detecting industrial and residential building roofs, respectively.

Advanced approaches have gone beyond simple classification, incorporating thermal imaging, semantic segmentation, and feature-matching algorithms. Mayer et al. [40] investigated CNN approach to detect thermal bridges on building rooftops recorded in panorama drone images from an updated dataset of Thermal Bridges on Building Rooftops (TBBRv2), containing 926 images with 6,927 annotations.

The images included RGB, thermal, and height information. Zahradník et al. [41] presented a method for detecting flat roof leaks using a combination of unmanned aerial vehicles and U-Net architecture. The model achieved an overall accuracy of 95% in segmenting the roof into different classes. The method was also able to identify damaged insulation with a high degree of accuracy. Yeh et al. [42] combined YOLOv3 with feature-matching algorithms to streamline rooftop detection, achieving speeds 13× faster than conventional manual methods. Similarly, Qin et al. [43] demonstrated a novel semantic segmentation approach for rooftops in dense urban areas, achieving an overall accuracy of 94.67%. Yudin et al. [44] utilized 6,400 pairs of aerial images and their corresponding ground truth masks, focusing on the segmentation of five categories of roof defects: hollows, swellings, folds, patches, and breaks. It evaluated multiple neural network architectures for segmentation tasks, including U-Net, DeepLabV3+, and HRNet+OCR, using Intersection over Union as a metric. The highest IoU was 0.44. The authors faced significant challenges due to the limited number of images in the dataset and the difficulties associated with imbalanced datasets in image segmentation tasks.

2.3 Theoretical Background

Computer vision is a field of study within artificial intelligence (AI) that focuses on enabling machines to intercept and extract information from images and videos, in a manner similar to human vision. It involves developing algorithms and techniques to extract meaningful information from visual inputs and make sense of the visual world [45].

Computer vision works by using image processing techniques to analyze and interpret visual data. This involves tasks such as image classification, object localization, object detection and image segmentation. These tasks are typically achieved through Machine Learning algorithms, such as Convolutional Neural Networks, trained on large datasets of labeled images. While visual information processing technology has existed for some time, much of the process required human intervention and was time consuming and error prone. For example, implementing a facial recognition system in the past required developers to manually tag thousands of images with key data points, such as the width of the nose bridge and the distance between the eyes. Automating these tasks required extensive computing power because image data is unstructured and complex for computers to organize. Vision applications were thus expensive and inaccessible to most organizations. Machine learning (ML) algorithms identify common patterns in images or videos and apply that knowledge to identify unknown images accurately. For example, if computers process millions of images of cars, they will begin to build up identity patterns that can accurately detect a vehicle in an image [46].

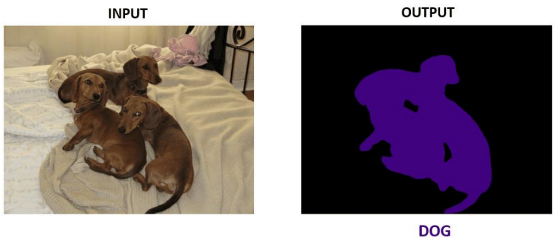
There are three stages of operation for a computer vision system [47]:

- Acquiring images: A computer vision system uses a camera or sensor to gather photos, movies, or other visual information.
- Process images: Acquired images are pre-processed through noise reduction, contrast adjustment, rescaling, and cropping, often automated by computer vision systems and hardware.
- Understanding images: It is the most important part of a computer vision system, where deep learning models or traditional image processing methods analyze and interpret the images.

Computer vision, a rapidly evolving field, leverages advanced algorithms to enable machines to interpret and understand images and video data, transforming industries such as healthcare, transportation, and entertainment. These algorithms fall into several key categories, each designed for specific tasks within the domain of computer vision:

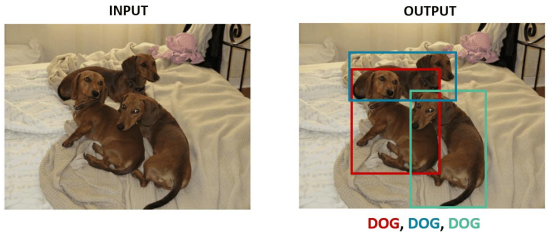
1. **Image Classification:** It is a fundamental task in computer vision that involves associating one or more labels with a given image. It enables systems to see an image and accurately assign it to a specific class [1].
2. **Object Localization:** This technique identifies and tracks items within an image, marking their spatial locations. It can be used for traffic monitoring in urban environments, human surveillance, and medical imaging [46].
3. **Object Detection:** Beyond localization, object detection involves classifying and identifying all objects within an image. It assigns a class to each object and draws bounding boxes around them [47].. Popular algorithms include You Only Look Once (YOLO) and Single Shot Detectors (SSD).
4. **Image Segmentation:** This task involves dividing an image into meaningful segments and labeling each fragment. Segmentation occurs at the pixel level, defining precise object boundaries. Depending on the use case, segmentation can be [47]:
    - Semantic Segmentation: Assigns a single label to all objects of the same class within an image, overlaying them with a segmentation mask. For example, an image of a city street can be segmented into vehicles, pedestrians, buildings, sidewalks, and so on. Objects of the same class are assigned the same pixel or color, as shown in Figure 2.1a [48]. Common models include U-Net and DeepLabv3+.

- Instance Segmentation: Extends semantic segmentation by assigning unique labels to individual objects, even within the same class. For instance, in the same city street image, each car or pedestrian would have a distinct label or color, as shown in Figure 2.1b [48]. The widely used model for this is Mask R-CNN.
- Panoptic Segmentation: Combines semantic and instance segmentation to provide a comprehensive understanding of the image.



(a) Semantic Segmentation. (Source)

(b) Instance Segmentation. (Source)



(c) Object Detection. (Source)

Figure 2.1: Comparison of Image Segmentation Types and Object Detection.

Computer vision (CV) has revolutionized various industries and enhanced daily life by automating tasks and improving efficiency. Its applications span multiple fields, including:

- Autonomous Vehicles: Computer vision enables vehicles to identify and track other vehicles, detect pedestrians, recognize road signs, and follow lane markings, ensuring safe operation without human intervention [1].

- Manufacturing: It automates quality checks, detects defects, improves assembly precision, and supports predictive maintenance to reduce downtime [47].
- Agriculture: It facilitates the identification of defects or irregularities in fruits, vegetables, and other crops, allowing for the removal of inferior items before reaching consumers [47].
- Healthcare: CV analyzes medical images (X-rays, CT scans, MRIs) to detect diseases and abnormalities, aiding doctors in accurate diagnoses and treatment planning [47].
- Spatial Analysis: In urban planning and architecture, CV powers spatial analysis, modeling 3D environments, analyzing pedestrian flows, and estimating retail space use [1].
- Facial recognition: Face recognition technology enables the identification or verification of individuals from a digital image or video frame. It is widely used in security systems, law enforcement, and personalized marketing [1].
- Content Organization: Computer vision systems automatically categorize and tag visual content, such as photos and videos, based on their content. This is particularly useful in digital asset management systems [1].
- Text extraction, or optical character recognition (OCR): It extracts text from images or videos, essential for digitizing printed documents, processing street signs in navigation systems, and real-time text analysis [1].
- Augmented reality (AR): It uses computer vision to superimpose digital information onto the real world, enhancing user experiences in gaming, retail, and education [1].
- Robotic automation: Enabling robots to perform tasks based on visual input, performing household tasks, supporting elderly care, or even managing inventory in warehouses [1].
- Sports: CV provides coaches and viewers with detailed analytics of players' movements and game strategies, offers automated highlights, real-time stats overlays, and enhanced interactivity in broadcasts [1].

These diverse applications demonstrate the transformative impact of computer vision across various fields. These advancements have been largely inspired by the human visual system. Researchers modeled the structure and functions of human vision, such as how the brain processes visual signals and recognizes objects, to develop neural networks and algorithms for image processing and pattern recognition. Human vision is a biologically-driven process that relies on the eyes to

capture light and the brain to interpret signals. It enables the intuitive recognition of objects and contexts through perception, recognition, and interpretation of visual information. Computer vision emulates these abilities using algorithms and neural networks trained on vast datasets. Unlike human vision, which excels in contextual understanding, computer vision processes images or videos with remarkable speed and precision, often surpassing human capabilities. However, it lacks the inherent ability to interpret deeper meanings beyond its training. This combination of human-inspired design and the computational power of machine learning drives the success of computer vision in real-world applications.

Whether it's identifying a tumor in a medical scan or detecting pedestrians in a self-driving car's camera feed, computer vision systems rely on processing raw image data, which is made up of millions of individual pixels. Pixels are the raw building blocks of an image. Every image consists of a set of pixels, with each pixel representing a color or intensity value at a specific location within the image. Pixels are typically represented in two ways: grayscale or color. In grayscale images, each pixel has a value between 0 and 255, where 0 represents black and 255 represents white, as shown in Figure 2.2. In color images, each pixel is represented in the RGB color space, defined by three channels: Red, Green, and Blue, with values ranging from 0 to 255. An RGB image can be thought of as consisting of three independent matrices, one for each color channel (Red, Green, and Blue). Each matrix has dimensions $W \times H$, where W is the width and H is the height. When these matrices are combined, they form a multi-dimensional array with the shape $W \times H \times D$, where D is the depth or number of channels (for RGB, D=3). It's important to note that the depth in an image (D) is different from the depth in a neural network [49].

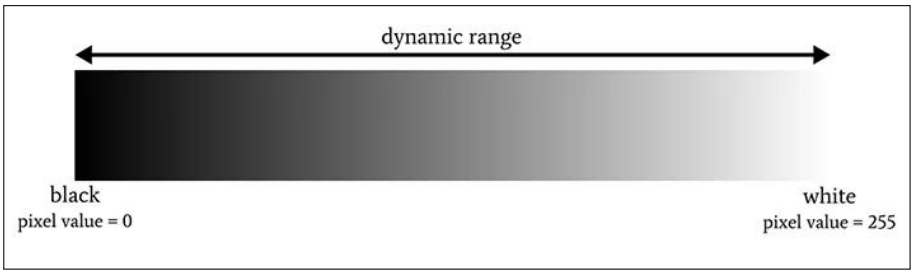


Figure 2.2: Image pixel values going from black (0) to white (255). (Source)

Before diving into Convolutional Neural Networks (CNNs), it is essential to first understand the basics of neural networks, which serve as the foundational models for deep learning. The first Neural Network model came from McCulloch and Pitts in 1943. This network was a binary classifier, capable of recognizing two different categories based on some input. The problem was that the weights used

to determine the class label for a given input needed to be manually tuned by a human. Then, in the 1950s the seminal Perceptron algorithm was published by Rosenblatt, this model could automatically learn the weights required to classify an input (no human intervention required). In fact, this automatic training procedure formed the basis of Stochastic Gradient Descent (SGD) which is still used to train very deep neural networks today [49].

Neural Networks are the core building blocks of deep learning systems. They are computational models that mimic the complex structure and functions of the human brain. They consist of interconnected nodes, or neurons, that process and learn from data, enabling tasks like pattern recognition and decision-making in machine learning [50]. Deep learning, a subset of machine learning, leverages multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain. These networks consist of multiple layers of interconnected nodes, each layer refines the output of the previous one, and optimizes the prediction or categorization. This progression of computations through the network is known as forward propagation. The output layer generates the final result, where the network makes its predictions or classifications. Backpropagation, another key component of neural network learning, calculates errors in predictions using algorithms like gradient descent and adjusts the model's weights and biases by propagating the errors backward through the layers. This iterative process, forward propagation to generate predictions and backpropagation to correct errors, neural networks learn to make accurate predictions by adjusting their parameters in response to the training data [51].

Artificial neural networks (ANNs) are inspired by the structure and operation of human neurons. In a fully connected ANN, there is an input layer, one or more hidden layers, and an output layer. The input layer receives data from external sources and passes it on to the hidden layers. Each neuron in these layers computes a weighted sum of inputs from the previous layer and applies an activation function, which determines the output passed to neurons in the next layer. This process continues through the network, layer by layer, until the output layer produces the final result. The strength of each connection, represented by a weight, is adjusted during training to improve the network's accuracy and performance [52].

The performance of neural networks in tasks like image processing is largely influenced by the choice of learning paradigm employed during training. Two primary paradigms are supervised and unsupervised learning. Supervised learning is learning through pre-labelled inputs, which act as targets. For each training example there will be a set of input values (vectors) and one or more associated designated output values. The goal of this form of training is to reduce the models overall classification error, through correct calculation of the output value of training example by training. Unsupervised learning differs in that the training set does not include any labels. Success is typically measured by the network's ability to

minimize or maximize a given cost function, depending on the task. It is important to note, however, that most image-focused pattern recognition tasks primarily rely on supervised learning for classification [53].

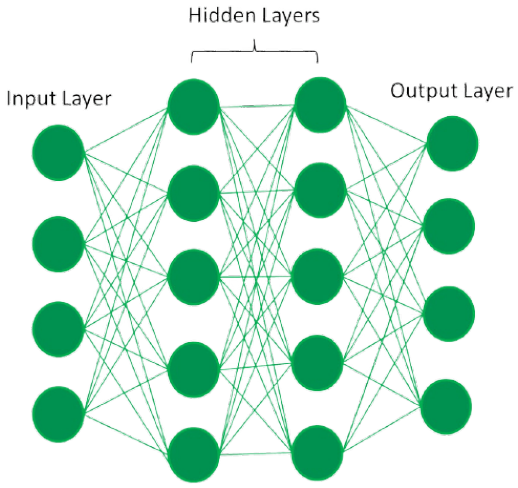


Figure 2.3: Fully Connected Artificial Neural Network.[52]

The widespread use of supervised learning in image-focused tasks has been largely driven by advancements in specific neural network architectures, particularly Convolutional Neural Networks (CNNs). Over the past decade, CNNs have achieved significant breakthroughs in a variety of fields, such as visual recognition, speech recognition, and natural language processing. CNNs have leveraged the rapid growth of annotated datasets and significant improvements in the computational power of Graphics Processing Units (GPUs). Research on CNNs has emerged swiftly, achieving state-of-the-art results across various tasks [33].

A Convolutional Neural Network, also known as ConvNet, is a specialized deep learning algorithm primarily designed for tasks that require object recognition, such as image classification, detection, and segmentation [54]. It consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The architecture of CNNs is inspired by the visual processing in the human brain, making them particularly well-suited for capturing hierarchical patterns and spatial dependencies within images [55]. In a CNN, each neuron still receives an input and performs an operation (e.g., a scalar product followed by a non-linear function), which forms the basis of many ANNs. From the raw input image vectors to the final output of the class score, the network still produces a single prediction score (the weight). The last layer contains the loss functions associated with the output classes, and many of the techniques developed for traditional ANNs are still applicable [56]. A significant limitation of traditional ANNs is that they struggle with the

computational complexity required for processing image data. A key difference between CNNs and traditional ANNs lies in the organization of neurons. In CNNs, neurons are arranged in three dimensions: height, width, and depth. The 'depth' here does not refer to the total number of layers in the CNN but rather to the third dimension of the activation volume. Unlike standard ANNs, the neurons in any given layer of a CNN only connect to a small region of the layer preceding it, making the structure more efficient for image-related tasks [56]. When the convolutional, pooling, and fully connected layers are stacked together, they form a complete CNN architecture.
The core functionality of a CNN can be broken down into four key areas [53]:

1. Like in an ANN, the input layer holds the pixel values of the image.

2. Convolutional Layers are the core building block of CNNs that perform most of the computational heavy lifting. They consist of a set of learnable filters (kernels) that are small spatially but extend through the full depth of the input. During the forward pass, each filter slides over the input volume, performing dot products between the filter and the input at each position. This produces a 2D activation map, representing the filter's response at each spatial location. These filters typically learn to detect specific visual features such as edges or color patterns. The activation maps are stacked along the depth dimension, creating the output volume. When dealing with high-dimensional inputs, each neuron is connected to only a local region of the input, defined by the receptive field, which is equal to the filter size. The connectivity along the depth axis is always equal to the depth of the input volume. The spatial arrangement and number of neurons in the output volume are controlled by three hyperparameters: depth, stride, and zero-padding [56]. The depth of the output volume corresponds to the number of filters used, with each filter learning to detect a different feature in the input. The depth of both the input array and the kernel array must always be equal. The stride defines the step size at which the filter moves across the input, with a stride of 1 moving one pixel at a time and a stride of 2 moving two pixels at a time. Zero-padding adds extra zeros around the input's border, helping to preserve spatial dimensions and control the output size [57]. The spatial size of the output volume is computed as follows:

$$\frac{W - F + 2 \cdot Z}{S + 1} \tag{2.1}$$

where W represents the input volume size (height $\times$ width $\times$ depth), F is the receptive field size, Z is the amount of zero padding, and S is the stride.

For example, a 7×7 input and a 3×3 filter, applying a stride of 1 and no

padding (pad = 0) produces a 5×5 output. If the stride is increased to 2, the output will be reduced to 3×3.

3. Pooling Layers are typically inserted between successive convolutional layers in a ConvNet architecture. Their main purpose is to reduce the spatial size of the representation, which decreases the number of parameters and computations in the network, thus controlling overfitting. Pooling layers operate independently on each depth slice of the input, resizing it spatially using operations such as max pooling. The depth dimension remains unchanged. They take in an input volume of size $W_1 \times H_1 \times D_1$ and require two hyperparameters: the filter size F and the stride S. The output volume size is given by $W_2 \times H_2 \times D_2$, where:

$$W_2 = \frac{W_1 - F}{S + 1}, \quad H_2 = \frac{H_1 - F}{S + 1}, \quad D_2 = D_1$$

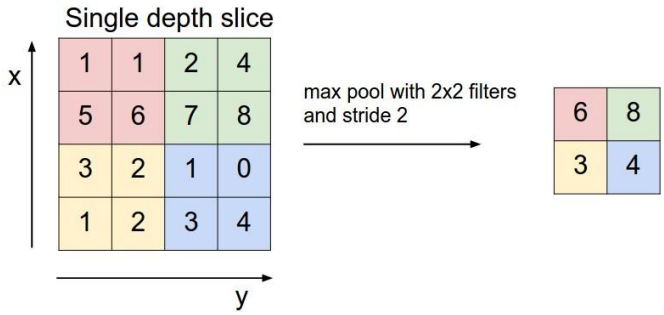


Figure 2.4: Max Pooling. (Source)

4. Fully-connected Layers: Neurons in a fully connected layer are connected to all activations in the previous layer, as in traditional neural networks. Their activations are computed using matrix multiplication, followed by a bias addition [56].

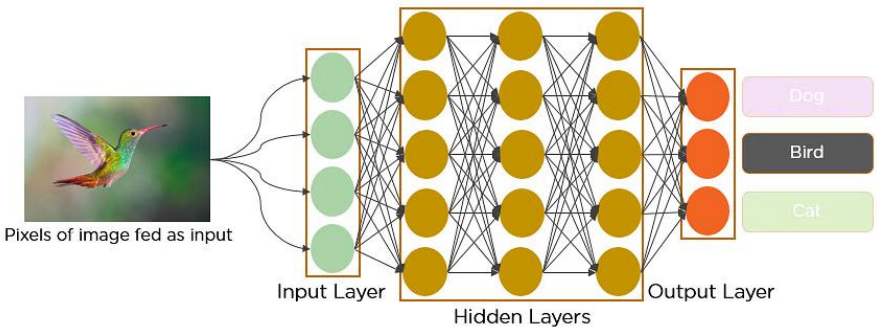


Figure 2.5: Convolutional Neural Network to identify the image of a bird. (Source)

Over the past decade, Convolutional Neural Networks (CNNS) have become the dominant technology in computer vision. Following the success of LeNet, a variety of CNN-based architectures have been developed, including AlexNet, GoogLeNet, VGGNet, Inception, Xception, ResNet, DenseNet, MobileNets, EfficientNets, and RegNet. This study explored and applied the following architectures:

1. VGG-16: It was implemented in the U-Net repository used in this work. In 2015, Simonyan and Zisserman investigated the effect of convolutional network depth on accuracy in large-scale image recognition setting. Their key contribution was evaluating networks with increasing depth, using very small 3×3 convolution filters. The image is passed through a series of convolutional layers with 3×3 filters, a fixed stride of 1, and padding of 1. Spatial pooling is performed with five max-pooling layers, each using a 2×2 window and a stride of 2. After the convolutional stack, three fully connected (FC) layers follow, with the first two having 4,096 channels and the third containing 1,000 channels (one for each class). The final layer is a softmax layer. ReLU activations are applied to all hidden layers, and the configuration of the fully connected layers is consistent across all networks [58].

2. Xception: It was implemented in the DeepLabv3+ repository. Xception was proposed by Chollet in 2017 as a novel deep CNN architecture inspired by Inception, but replacing Inception modules with depthwise separable convolutions. Xception slightly outperforms InceptionV3 on the ImageNet dataset and significantly outperforms it on larger image classification datasets, including one with 350 million images and 17,000 classes. The Xception architecture consists of 36 convolutional layers organized into 14 modules. Each module incorporates linear residual connections, except for the first and last modules. The data passes through the entry flow, then the middle flow (repeated eight times), and finally through the exit flow [59].

3. ResNet-50: It was implemented in both U-Net and Mask R-CNN repositories. It was introduced by He et al. in 2015 [60]. A variant of the ResNet architecture, the "50" refers to the network's 50 layers. A key feature of ResNet-50 is the use of skip (residual) connections, which help overcome the vanishing gradient problem in deep networks. These connections allow information to bypass certain layers, enabling deeper architectures without losing the ability to learn effectively. In ResNet-50, there are two types of residual blocks: identity blocks, where the input is passed through convolutional layers and added back to the output, and convolutional blocks, which use a 1×1 convolution to reduce the number of filters before the 3×3 convolution, then add the input back to the output [61].

4. MobileNetV2: It was implemented in the DeepLabv3+ repository. It was

proposed by Sandler et al. in 2018. Its architecture features an inverted residual structure, where the input and output of the residual block are small bottleneck layers, in contrast to traditional residual models. The model uses efficient depthwise convolutions to filter features in an intermediate expansion layer. MobileNetV2 starts with a fully convolutional layer containing 32 filters, followed by 19 residual bottleneck layers. With a model size between 1.7M and 6.9M parameters, MobileNetV2 is highly memory-efficient, making it ideal for mobile applications. Its linear bottleneck structure offers fewer activations but maintains efficiency by operating in a linear regime with appropriate biases and scaling.[62]

5. CSPDarknet53: This CNN architecture is used as the backbone for object detection with DarkNet-53 and serves as the main backbone for YOLOv8. It employs a Cross Stage Partial Network (CSPNet) strategy, which partitions the feature map of the base layer into two parts and merges them through a cross-stage hierarchy. The use of a split and merge strategy allows for more gradient flow through the network. CSPNet was proposed by Wang et al. in 2019 [63] to mitigate the problem of heavy inference computations from the network architecture perspective.

Activation functions play a crucial role in the operation of neural networks by introducing non-linearity to the model, enabling the network to learn complex mappings between inputs and outputs. Among the most widely adopted activation functions are Sigmoid, ReLU, Tanh, and Leaky ReLU. These functions determine how the weighted sum of inputs is transformed and passed to the next layer, significantly influencing the network's ability to train effectively and generalize well.

The Sigmoid function maps input values into the range (0, 1). This property makes it particularly suitable for binary classification problems. Mathematically, it is expressed as:

$$\text{Sigmoid} = \frac{1}{1 + e^{-z}} \tag{2.2}$$

where "e" refers to the base of the natural algorithm, and "z" is the input value (weighted sum) [64].

Rectified Linear Unit (ReLU) is the most commonly used activation function. It outputs the input value if positive and zero otherwise, and is defined as:

$$\text{ReLU} = max(0, z) \tag{2.3}$$

ReLU introduces sparsity by activating only a portion of the neurons, which helps in representing high-dimensional data efficiently. It also mitigates the vanishing gradient problem, enabling faster training. However, it is prone to the "dying

ReLU" issue, where neurons can become inactive if their outputs consistently fall below zero [64].

The Hyperbolic Tangent (Tanh) transforms the input value between -1 and 1, offering zero-centered output, which can facilitate optimization. It is represented as:

$$\text{Tanh} = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{2.4}$$

Tanh is often used in classification tasks where inputs are symmetric around zero. However, like Sigmoid, it suffers from the vanishing gradient problem when neurons saturate, particularly for large input values [64].

Leaky ReLU is a variant of ReLU that addresses the "dying ReLU" problem by allowing a small, non-zero gradient for negative input values. Mathematically, it is given as:

$$\text{LeakyReLU} = max(0.01z, z) \tag{2.5}$$

Here, $0.01z$ is the slope for negative inputs, ensuring that all neurons can contribute to learning during backpropagation. Leaky ReLU is particularly effective when dealing with high-dimensional data [64].

These activation functions are selected based on the specific requirements of the model, such as the task type, data characteristics, and network depth. Together, they enhance the expressive power of neural networks by enabling the capture of complex, non-linear patterns in the data.

Deep Learning is extremely data-hungry and demands an extensive amount of data to achieve high performance. In other words, DL models often struggle when data is limited. However, efforts have been made to train models with less data, and this is where transfer learning comes in as a technique to address these challenges. Transfer Learning is a powerful technique that leverages knowledge gained from one task to improve performance on another task. It is particularly useful in deep learning because training deep neural networks from scratch can be computationally expensive and time-consuming. In this study, transfer learning was employed to enhance the performance of neural network models. By reusing pretrained architectures, such as VGG-16, ResNet-50, and MobileNetV2, the models could focus on learning task-specific features, reducing both training time and data requirements [65].

There are two approaches to perform transfer learning: Feature Extraction and Fine-tuning. In feature extraction, the weights of the convolutional layers are frozen, preserving their learned weights. A new fully connected layer is added at the end of the network, and only this fully connected layer is trained. This method is particularly effective when the source and target tasks are closely related. In fine-tuning, the last few convolutional layers of the pretrained model are unfrozen, allowing them to adapt to the new task. A new fully connected layer is also

added and trained. This method is useful when the source and target tasks differ significantly [65].

This work explored the potential of deep learning by implementing four CNN architectures tailored to different tasks:

1. **DeepLabv3+**: It was introduced by Chen et al. in 2018. It extended DeepLabv3 by adding an encoder-decoder structure with Atrous Separable Convolutions to refine the segmentation results especially along object boundaries. The authors further explored the Xception model and applied the depthwise separable convolution to both Atrous Spatial Pyramid Pooling and decoder modules, resulting in a faster and stronger encoder-decoder network [66].

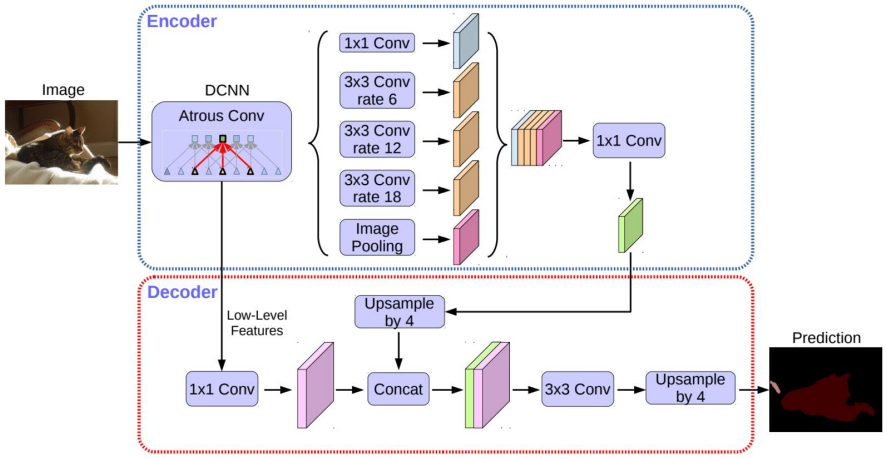


Figure 2.6: Encoder-decoder structure of DeepLabv3+ [66].

2. **U-Net**: It was proposed by Ronneberger et al. in 2015 for Biomedical Image Segmentation. It consists of a contracting path (left side) and an expansive path (right side), see Figure 2.7. The contracting path consists of the repeated application of two 3×3 convolutions, each followed by a rectifed linear unit (ReLU). The expansive path consists of an upsampling of the feature map followed by a 2×2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3×3 convolutions, each followed by a ReLU. In total the network has 23 convolutional layers [67].

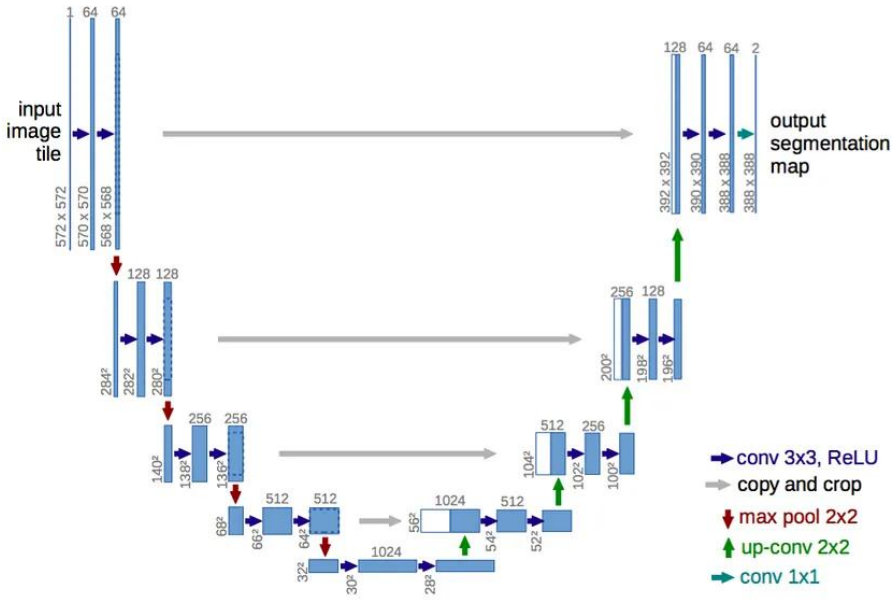


Figure 2.7: U-net Architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. (Source)

3. **Mask R-CNN**: It was introduced by He et al. in 2017. It extended Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. Faster R-CNN consists of two stages. The first stage, called a Region Proposal Network (RPN), proposes candidate object bounding boxes. The second stage extracts features using RoIPool from each candidate box and performs classification and bounding-box regression. Mask R-CNN adopts the same two-stage procedure, with an identical first stage. In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI [68].

4. **You Only Look Once (YOLO)**: It was presented by Redmon et al. in 2015. The YOLO series has revolutionized the field by framing object detection as a single regression problem, where a CNN processes an entire image in one pass to predict bounding boxes and class probabilities. This approach has marked a departure from traditional multi-stage detection methods, offering significant gains in speed and efficiency. The architecture of YOLOv8 is structured around three core components: Backbone, Neck, and Head. The backbone is an advanced version of CSPDarknet architecture. The head module is responsible for generating the final predictions, including bounding box coordinates, object

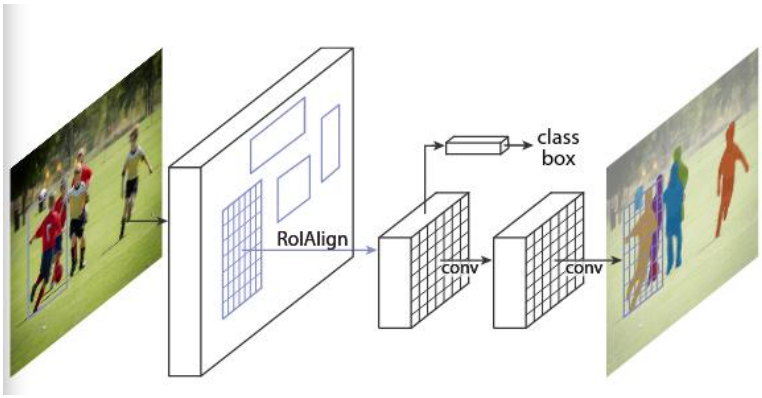


Figure 2.8: Mask R-CNN framework for Instance Segmentation.

confidence scores, and class labels. The neck module refines and fuses the multi-scale features extracted by the backbone. The YOLOv8 architecture has introduced five distinct models: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x [69].

After understanding the CNN architectures and transfer learning techniques applied in this work, the next crucial step involves selecting and tuning hyperparameters, which significantly influence the model's performance. In ML/DL, training a model involves selecting the optimal hyperparameters. These parameters govern the learning process and affect how the model learns its internal parameters during training. As a machine learning engineer, hyperparameter values can be chosen and configured prior to starting the model training process. Hyperparameter tuning is the process of selecting the optimal values for a machine learning model's hyperparameters. The goal of hyperparameter tuning is to find the values that lead to the best performance on a given task. Here are some common examples [71]:

- Train-val-test split ratio.
- Epochs: It represents the number of times the entire training dataset is passed through the model during training.
- Batch size: This refers to the number of training examples used in each iteration of the optimization algorithm [72].
- Learning rate: It controls the step size taken by the optimizer during each iteration of training. Too small learning rate can result in slow convergence, while too large learning rate can lead to instability and divergence.
- Number of layers: It determines the depth of the model, which can have a significant impact on its complexity and learning ability.

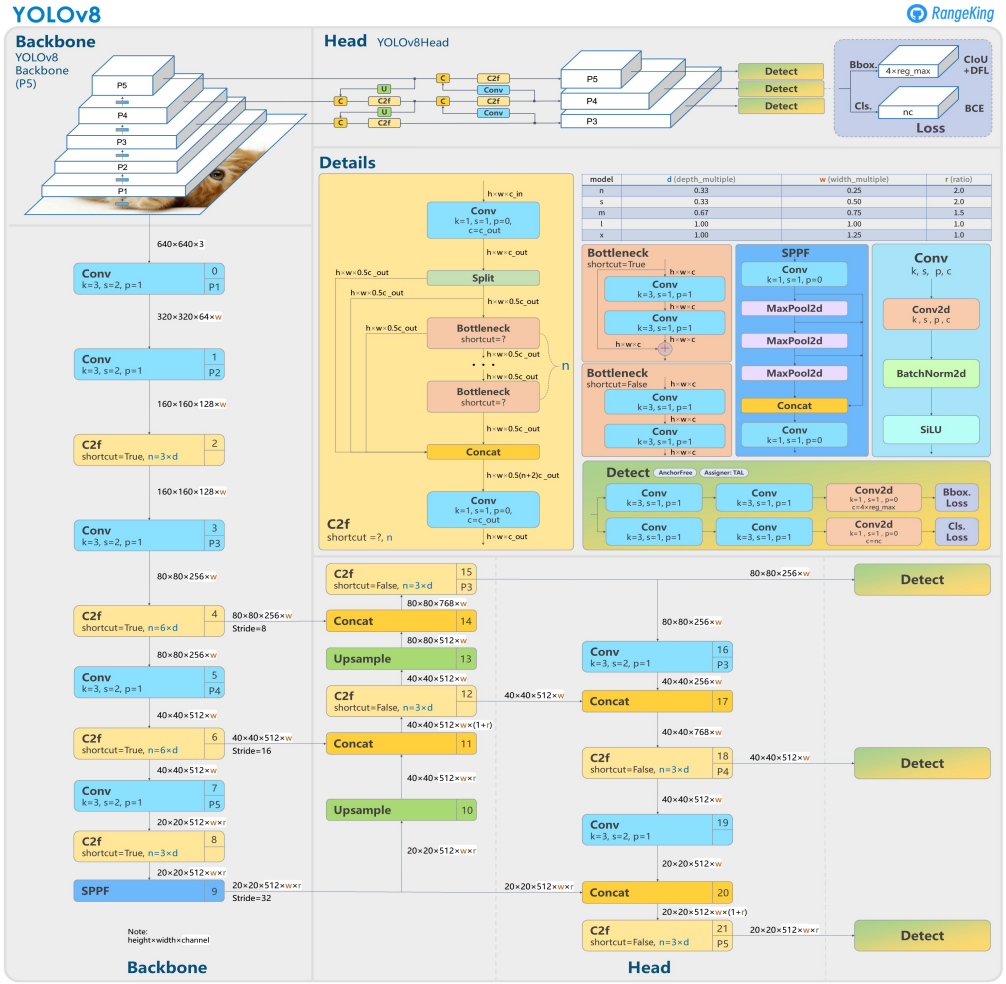


Figure 2.9: YOLOv8 Architecture.[70]

- Optimizer: The algorithm used to minimize the loss function by updating the model weights. Common optimizers include Adam, Stochastic Gradient Descent (SGD), and RMSprop [72].
- Learning Rate Decay: The technique of reducing the learning rate as the training progresses [72].
- Gradient Clipping: A technique to prevent exploding gradients in deep neural networks by limiting the values of gradients to a small range [72].
- Choice of activation function in a neural network layer (e.g. Sigmoid, ReLU).
- The cost or loss function the model will use.

- The drop-out rate in NN (dropout probability).
- Momentum: It helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9.
- Kernel or filter size in convolutional layers.
- Pooling size.

Deep Learning has become the dominant technology for solving unstructured data problems. One of the critical components of deep learning is the selection of a loss function. Loss functions measure how effectively a model can approximate the desired output. Choosing the appropriate loss function is crucial for achieving success in deep learning tasks. However, with many options available, it can be challenging to determine the most suitable one for a specific task [73].

In computer vision, many loss functions exist, but only a few were applied in this work. Three loss functions were implemented in DeepLabv3+ and U-Net which are: dice loss, focal loss, and cross-entropy (CE) loss. Their definitions are provided below:

- Cross-entropy Loss is commonly used in segmentation tasks. It calculates the dissimilarity between the predicted probabilities and the actual labels for each pixel. In multi-class segmentation, the cross-entropy loss is defined as the negative log-likelihood, which is a measure of how well the predicted probability matches the true class for each pixel [73].
- Dice Loss is based on the Dice Similarity Coefficient (Sørensen-Dice index), it quantifies the similarity between predicted and ground truth segmentation masks. This coefficient is particularly effective for imbalanced datasets where the regions of interest are significantly smaller than the background [73].
- Focal loss addresses the challenge of class imbalance, it has been adapted effectively for use in semantic segmentation. Focal loss modifies the standard cross-entropy loss equation to emphasize hard-to-classify cases. This enhances the model's focus on misclassified examples, improving balance between positive and negative examples [73].

For Mask R-CNN, a multi task loss function has been utilized [74]:

$$L = L_{\text{cls}} + L_{\text{box}} + L_{\text{mask}} \tag{2.6}$$

where, L_{cls} is the classification loss, which is the log loss over two classes (object vs. not object), computed with a cross-entropy loss function [74]. L_{box} is the bounding

box loss, calculated using a regression loss, such as smooth L1 [74]. L_{mask} is the mask loss, defined as the average binary cross-entropy loss [68].

YOLOv8 employs a tailored loss function comprising three main components: Focal Loss for Classification, IoU Loss for Localization, and Objectness Loss. IoU Loss enhances the accuracy of bounding box predictions, refining the model's ability to precisely localize objects within the image. The Objectness Loss ensures that the model focuses on regions of the image that are likely to contain objects, thereby improving its overall detection capability [75].

The choice of an appropriate loss function is crucial for effective model training, but once the model is trained and the loss function has been minimized, it is essential to evaluate the model's performance on unseen data. This is where performance metrics come into play. These metrics assess how well the model generalizes and makes accurate predictions. The metrics used in this work include Intersection over Union (IoU), Precision, and Recall, which provide insight into the model's ability to accurately detect and segment objects in the given task. They are defined as follows:

1. IoU is defined as the area of intersection between the predicted segmentation map and the ground truth, divided by their area of union [73].

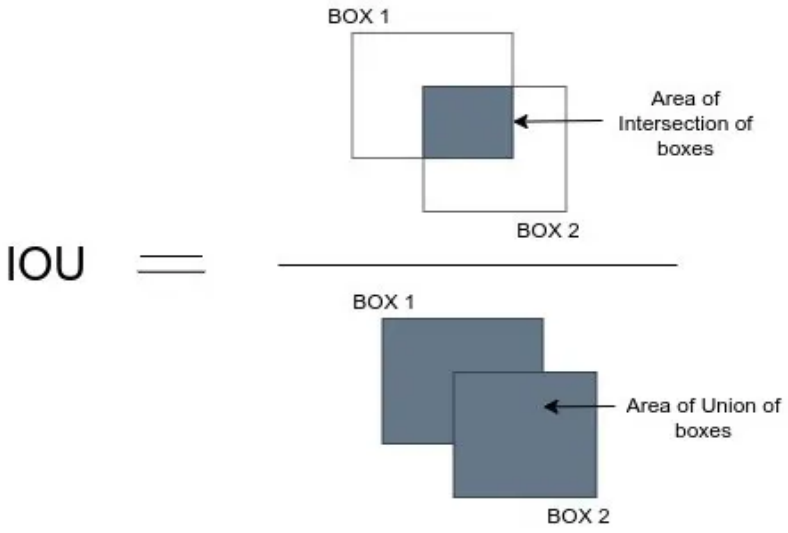


Figure 2.10: Diagrammatic Representation of the IoU formula [76].

2. Precision measures the accuracy of positive predictions made by a model [73].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.7}$$

3. Recall is a measure of a model's ability to correctly identify positive instances among all actual positives in the dataset [73]. It is calculated as:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.8}$$

where:

- True Positives (TP) are the cases where the model correctly predicts the positive class [73].
- True Negatives (TN) are the cases where the model correctly predicts the negative class [73].
- False Positives (FP), also known as Type I errors, occur when the instances are actually negative but are wrongly labeled by the model as positive. This is sometimes called "false alarm" [73].
- False Negatives (FN), also known as Type II errors, occur when the instances are actually positive but are mistakenly labeled by the model as negative. This is sometimes referred to as a "miss" [73].

2.4 Methodology

2.4.1 Dataset Preparation

This section describes the complete methodology used to develop an accurate approach for detecting and classifying rooftops from UAV images. The process has included several key phases: understanding and preparing the data, exploring CNN model architectures, creating a synthetic dataset, and conducting training, validation, and testing. Finally, a fine-tuning phase has been conducted to adapt the trained models to real-world UAV images and evaluate their performance.

A 3D model of a real industrial facility has been imported into a graphic engine software called Unreal Engine. Using a MATLAB/Simulink script that interfaces with Unreal Engine, a synthetic dataset has been created through a simulation, producing realistic synthetic images derived from the 3D model. Key parameters such as drone height, sweep angle, image dimensions, focal length, Ground Sample Distance (GSD), UAV sample time, and speed have been specified through a MATLAB script, ensuring precise control over the dataset generation process. These parameters were essential for fine-tuning the simulation to obtain an accurate dataset, and specific parameters were carefully assigned for each map to control the simulation. The roof of the industrial facility contained many different objects like pipelines, solar panels and others, these objects were assigned specific number and color for the segmentation, which they were defined in a different Matlab script. Then, the image of the entire roof was augmented with additional meshes and roof textures to expand the dataset and to provide more roofs for the networks to train on and learn from. Since the goal of this work is to detect two types of roofs: Flat roofs and gabled roofs, the added textures included both types of roofs. But the existence of several objects on the rooftops created major complexity and posed significant challengies. Due to these complexities, it was decided to begin with a simpler dataset in which rooftops were more apparent and distinct. To achieve this, a less complex map was used to offer clearer rooftop views, facilitating the refinement of the model's accuracy before applying it to more challenging datasets. A map called USCityBlock was chosen, which is available in the MathWorks website, see figure 2.11.

The simulation produced a .mat file, which contained 2 key elements: one named "images" for storing collected images, and another named "labels" to store the corresponding graound truth labels. Both were initially 1080×1080 pixels but they were subsequently extracted in .png format and cropped to a resolution of 448×448 pixels, a suitable size for CNN input, enabling the models to process a uniform input format.

The following parameters are the main parameters that were specified in the MATLAB file to control the simulation.

- Drone Height: The altitude at which the drone operates, affecting the field of view and resolution of acquired images.
- Sweep Angle: The angle at which the drone captures imagery, determining coverage and overlap between subsequent shots.
- Image Width and Height: The dimensions of the collected images, influencing the level of detail in the rooftop analysis.
- Focal length: The distance between the camera lens and the image sensor.
- GSD: Ground Sample Distance is the distance between the centers of two consecutive pixels on the ground, and it's a measure of the spatial resolution of an image. The formula to calculate GSD is:

$$\text{GSD} = \frac{\text{Flight Altitude (A)} \times \text{Sensor Pixel Size (P)}}{\text{Focal Length (F)}}$$

  where:

$$\begin{aligned} A &= \text{Flight Altitude (in meters)}, \\ P &= \text{Sensor Pixel Size (in millimeters)}, \\ F &= \text{Focal Length (in millimeters)}. \end{aligned}$$

- UAV SampleTime: The interval at which the system collects data, updates its state, or executes commands.
- Nth Frame Interval: Set to capture every n-th Frame, this determined the frequency of image capture during the flight.
- Look Ahead Distance: The speed of the drone.
- WpSamplingFactor: The distance in meters desired in order to interpolate waypoints.

The dataset was also preprocessed to match the input requirements of the chosen CNN architectures. Label conversion was performed to ensure compatibility with each model's expected input format. The images generated by the simulations were immediately ready to be passed to the networks, but the ground truth labels needed to be modified. Since each map contained many segmented objects, all of them were represented in the ground truth labels. Given that the focus of this work is to detect only rooftops, it was necessary to convert the all the pixels that don't represent a rooftop to zero (black color), classifying them as background.

Both datasets contained 4,380 images along with their corresponding ground truth labels, same numbers of images were used for both maps to maintain consistency and ensure reliable comparisons. Initially, 108 images were randomly selected for the test dataset. The remaining images were divided into 90% for training (3,780 images) and 10% for validation (420 images), ensuring consistent data distribution across the training and validation subsets. This split ensured that the networks had enough data for learning while keeping the test set unseen for a more accurate performance evaluation. The facility dataset consisted of two subsets: one of synthetic images and the other of real images.

Four Convolutional Neural Network (CNN) architectures were selected for this study: Mask R-CNN [77], DeepLabV3+ [78], U-Net [79], and YOLOv8 [80]. The architectural design of each model was examined to determine its suitability for the task, with particular attention given to input requirements, layer composition, and output compatibility. A thorough review of each architecture's segmentation capabilities allowed for the customization of parameters to fit the specifics of this application.

For both DeepLabv3+ and U-Net, dataset preparation was simpler compared to Mask R-CNN and YOLOv8. As mentioned earlier, in the ground truth labels, all non-rooftop pixels were assigned a value of zero, while rooftop pixels were assigned a value of 1 for the urban map, while for the facility map, gable rooftop pixels were labeled as 1, and flat rooftop pixels as 2. These labeled images were then passed along with the corresponding images to the models.

For Mask R-CNN, the dataset preparation was more complex, as the network performs both instance segmentation and object detection. This required detailed annotations for each rooftop, including bounding boxes, segmentation masks, and class labels. Mask R-CNN has several datasets, each with its own specific structure for organizing images and annotations. The dataset used in this work was Common Objects in Context (COCO) [81]. To adapt the dataset for this task, the ground truth folders for training, validation, and testing were converted into JSON files. These files contained all necessary details about the rooftop pixels, ensuring that each rooftop was properly annotated for Mask R-CNN to process.

For YOLOv8, the dataset was structured for both object detection and segmentation tasks. Each image was paired with a text file containing bounding box annotations for each rooftop in the image. Each line in the text file represented a single bounding box, including its coordinates and class label, all following the YOLO format, while images without any rooftop had empty text files.

$$< class\_id >< center\_x >< center\_y >< width >< height > \tag{2.9}$$

To achieve a dependable and uniform comparison, same hyperparameters were used across all four networks for boths maps, providing uniform feature extraction

and maintaining comparable conditions throughout the training process, and several runs were made to fine-tune the networks (this is discussed in the following parts).

These hyperparameters were chosen to train the final phase of the project. Before this phase, multiple training sessions were conducted to understand each network and the parameters that the GPU can work with. This approach helped in selecting common values for all parameters, as all four networks needed to be trained with the same dataset and hyperparameter values. Some models performed better with different parameter values, but when those values were used with other networks, the GPU could not complete the full training. For instance, Mask R-CNN showed improved results with a batch size of 16. However, DeepLabV3+ and U-Net could not proceed with a batch size of 16 due to GPU RAM limitations. Additionally, DeepLabV3+ and U-Net achieved better results even with fewer than 20 epochs, reaching convergence sooner, so there was no need to increase the number of epochs beyond 20 (which was the initial value for the final phase). Twenty was set as the maximum number of epochs because less than 20 led to poor results for YOLOv8 and Mask R-CNN. Therefore, some hyperparameters, such as epochs and batch size, were kept fixed across all training sessions and comparisons, while others were varied in each training. Comparisons will be made between all networks according to the evaluation metrics used. However, in this section, a comparison will be made for each network individually, examining how each one performed when fine-tuning the hyperparameters.

In DeepLabv3+ and U-Net, the settings for Dice loss and focal loss can be adjusted to true or false. If focal loss is set to false, the model uses Cross-entropy loss instead.

2.4.2 Model Customization and Backbone Integration

For consistency and fair comparison, a single backbone, ResNet-50, has been used across all four models: DeepLabv3+, U-Net, Mask R-CNN, and YOLOv8. This choice was made to minimize the effect of different backbones on the performance and to ensure that any differences in performance were solely attributed to the architectures of the models themselves, rather than variations in backbone performance. While U-Net and Mask R-CNN repositories had built-in support for ResNet-50, modifications were required for DeepLabv3+ and YOLOv8, as their implementations were based on different backbone architectures, such as Xception and MobileNetV2 for DeepLabv3+ and custom definitions for YOLOv8.

The process of integrating ResNet-50 into DeepLabv3+ involved several steps. Since the original DeepLabv3+ implementation in the repository used Xception and MobileNetV2 as backbones, the code and parameters had to be adjusted to accommodate ResNet-50. The function definitions were carefully modified, ensuring

compatibility with ResNet-50's structure and parameters. Specifically, the ResNet-50 function was taken from the U-Net repository, and then it was implemented, and integrated into DeepLabv3+. Xception and MobileNetV2 have a different internal architecture compared to ResNet-50, which required the code to be carefully adjusted. The modification process was crucial for ensuring that DeepLabv3+ could now leverage ResNet-50 for feature extraction while maintaining the rest of the segmentation architecture unchanged. This adjustment was important for achieving consistency in backbone usage across all models.

Integrating ResNet-50 into YOLOv8 presented a more complex challenge, particularly due to the YAML configuration file used by the model. YOLOv8 uses a special YAML file to define its backbone and head configuration, making the integration of a new backbone slightly more complicated. In the original YOLOv8 configuration, the backbone was defined using pre-configured functions and custom layers that were tailored to different backbones. Modifying the YAML file required a deep understanding of the architecture and specific code adjustments to ensure that the new ResNet-50 backbone would be correctly integrated. The configuration file was edited to include the appropriate parameters for ResNet-50, ensuring that the backbone was properly initialized and linked to the rest of the YOLOv8 network.

2.4.3 IoU Implementation for YOLOv8 and Mask R-CNN

As part of the evaluation process, Intersection over Union (IoU), Precision, and Recall were chosen as the primary metrics to assess and compare the performance of the four deep learning models. While the DeepLabv3+ and U-Net repositories already included built-in functions to compute IoU for each class as well as mean IoU (mIoU), both YOLOv8 and Mask R-CNN required custom implementations for this evaluation.

The YOLOv8 repository has not provided a built-in function for computing IoU directly from the model's predictions. Therefore, a custom function has been developed to calculate the IoU for each class by comparing the predicted bounding boxes with the ground truth (GT) annotations. The predicted bounding boxes have been saved during the validation phase by setting the argument 'save_txt' to True. These predicted boxes have been compared with the ground truth bounding boxes by calculating the intersection area (overlap) between the predicted and GT boxes. The union has been calculated by adding the areas of both the predicted and GT boxes and subtracting the intersection area to avoid double-counting.

Mask R-CNN outputs instance segmentation results, where each detected instance (rooftop) has been associated with predicted bounding boxes, represented as 'pred_boxes'. These predicted bounding boxes, along with the ground truth (GT) bounding boxes (stored in the JSON annotation files), have been used to compute IoU. The area of intersection has been calculated by determining the overlapping

area between the predicted and ground truth boxes, while the union has been computed by summing the areas of the predicted and GT boxes and subtracting the intersection area.

2.4.4 Urban Synthetic Map

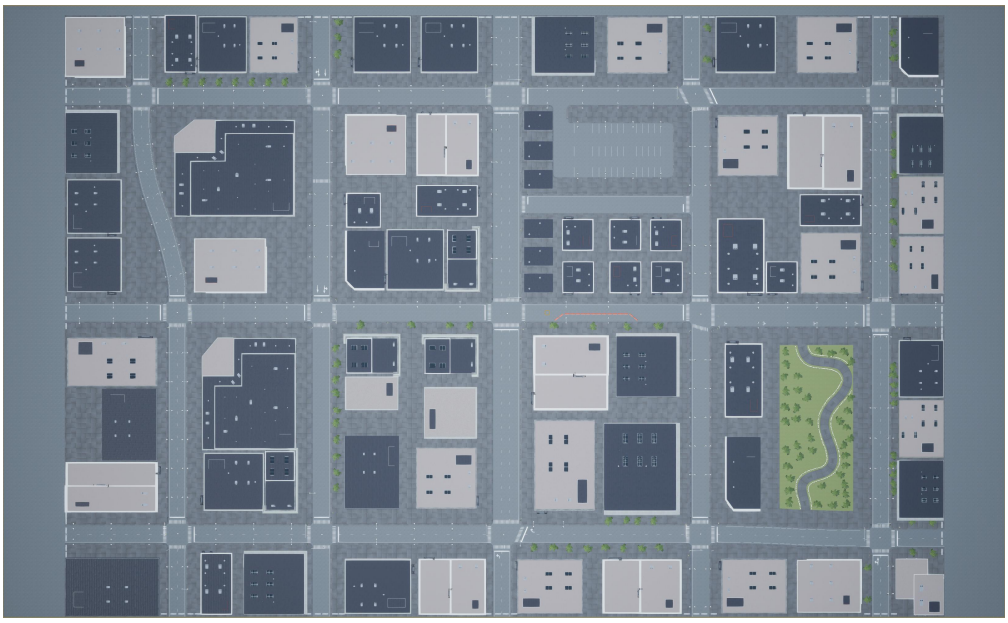


Figure 2.11: Urban Synthetic Map.(Source)

As previously mentioned, this work aims to detect two types of rooftops: flat and gabled, but this map contains only one type of rooftop which is flat (a further contributing to its simpicity). This reduced complexity allowed for better initial training, as a fundamental parameter in training any network is to clearly define the number of classes.

As seen in figure 2.11, the map consists of various objects, such as: buildings, roads, lanes, and other features, each with its own segmentaion color. This map was imported into the Unreal Engine, and was also downloaded as an image and then inserted into the matlab code to initiate the simulation and create the needed dataset.

The parameters used in the matlab script to control the simulation of the urban synthetic map were as follows:

Drone Height	130 m
Sweep Angle	45°
Image Width and Height	1080x1080
Focal length	35 mm
GSD	2.749 cm/pixel
UAV SampleTime	0.1 s
Nth Frame Interval	20
Look Ahead Distance	5 cm/s
WpSamplingFactor	8 m

Table 2.1: Simulation Parameters.

The simulation was initially conducted by selecting the entire area, setting the image size to 1920x1080 pixels. Although the simulation completed successfully, a "out of memory" error encountered while saving the images and labels, which was caused by a RAM limitation due to the large amount of data being processed. To fix this issue, the simulation was rerun using a smaller area, but after two hours, it crashed again due to the same error. The same problem occurred again in the third simulation, even though a smaller area was selected. The fourth simulation was then conducted on the same area as the third, but the image size was reduced to 1080x1080 pixels, this allowed the images and labels to be saved successfully. Then, fifth and sixth simulations were performed on the remaining parts of the area with the new image size, while keeping the same image size.

Since the 1080x1080 resolution produces significantly less data compared to the 1920x1080 resolution, the demand on the RAM was reduced, allowing the simulation to process a smaller volume of data overall. The reduction in image size directly impacts the amount of memory required, as fewer pixels need to be stored and processed at each step of the simulation. This reduction was essential to overcoming the memory limitations encountered during the previous simulations. By choosing a lower resolution, the data generated became more manageable for the system, making it less likely to exceed the available memory resources. The adjustment was not only crucial in preventing further memory-related issues, but it also optimized the simulation performance while still providing adequate image quality for the rooftop detection task.

By splitting the area into smaller segments and reducing the resolution of the images, the memory limitations were overcome, and the simulations were completed successfully, producing the required dataset.

Multiple training sessions were conducted for each network, and as previously mentioned, the results of these various trainings will be compared based on each individual network. In other words, for each network, several training runs were performed where hyperparameters were fine-tuned to determine the optimal settings

for achieving the best results. All runs for each network were conducted on the same dataset, using the same number of training, validation, and testing images to ensure a fair and reliable comparison.

DeepLabv3+

The results obtained from training this model for the detection of rooftops are summarized in Table 2.2. The table includes various configurations of the model, employing two backbone architectures: MobileNetV2 and Xception, and two optimization strategies: SGD (Stochastic Gradient Descent) and Adam. The performance metrics reported include Intersection over Union (IoU), Recall (R), and Precision (P), which are essential for evaluating the effectiveness of the model in accurately identifying rooftops in images.

The best-performing configuration was MobileNetV2 with Adam, 10 epochs, and a learning rate of 5e-4 (Run 2). This model achieved excellent performance across all metrics with a high IoU, precision, and recall, it also provided the best compromise between all metrics and training time. On the other hand, Xception struggled with just 10 epochs across both SGD and Adam. It required 30 epochs to converge and show significant improvement over the 10 epochs version. However, despite the improvement with 30 epochs, Xception still did not surpass MobileNetV2 in terms of overall efficiency, highlighting that MobileNetV2 achieved strong performance in fewer epochs, making it a more resource efficient choice for this task.

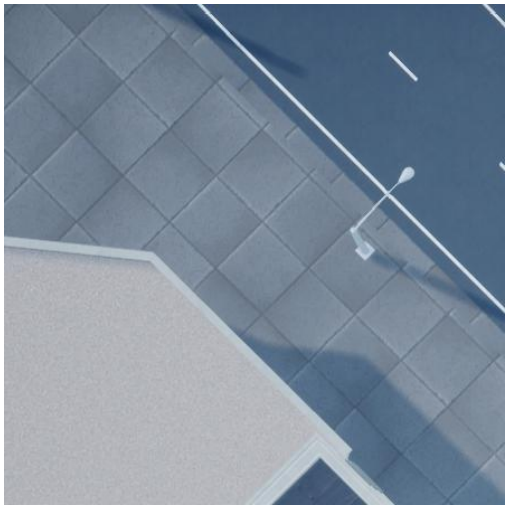
Run	Backbone	Epochs	BS	Opt	lr	IoU	P	R
1	MobileNetV2	10	8	SGD	7e-3	97.77%	98.57%	99.17%
2	MobileNetV2	10	8	Adam	5e-4	97.97%	98.67%	99.28%
3	Xception	10	8	SGD	7e-3	82.31%	93.77%	87.07%
4	Xception	20	8	SGD	7e-3	93.72%	95.99%	97.54%
5	Xception	30	8	SGD	7e-3	96.15%	97.33%	98.76%
6	Xception	10	8	Adam	7e-3	81.1%	94.05%	85.48%
7	Xception	10	8	Adam	5e-4	83.74%	94.51%	88.02%
8	Xception	20	8	Adam	5e-4	95.47%	97.08%	98.29%

Table 2.2: DeepLabv3+ Training Results

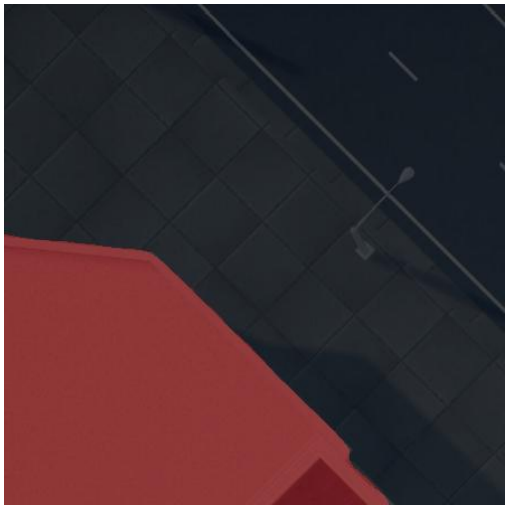
Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate, R = Recall, P = Precision

U-Net

Table 2.3 compares the performance of U-Net when altering specific hyperparameters, including the backbone (ResNet-50 vs. VGG-16), optimizer (SGD vs. Adam).



(a) Input Image.



(b) Predicted mask from DeepLabv3+.

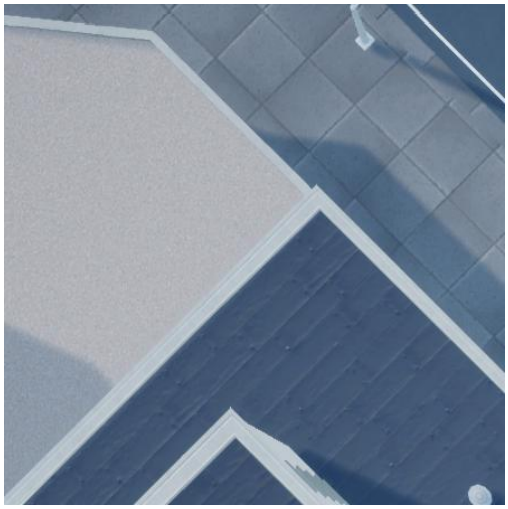
Overall, the results demonstrate that the choice of backbone and optimizer significantly affects the model's ability to perform the given task. ResNet-50 paired with Adam shows a slight improvement in performance compared to SGD, achieving an IoU of 98.41% compared to 98.12%, with marginal increases in both Precision and Recall. Similarly, for the VGG-16 backbone, Adam outperforms SGD by a more substantial margin, reaching an IoU of 98.85% versus 97.94%, accompanied by higher Precision and Recall. This indicates that Adam consistently provides a more effective optimization strategy than SGD, particularly when used with VGG-16. Furthermore, the results reveal that VGG-16, as a backbone, outperforms ResNet-50 across all metrics regardless of the optimizer, with the combination of VGG-16 and Adam delivering the highest scores overall. This suggests that VGG-16's architecture is better suited for feature extraction in this task, and its performance is further enhanced by Adam's adaptive optimization, making this pairing the most effective configuration.

During the third run, at epoch 6, a sudden spike in the training loss was observed. The loss steadily decreased from 0.1706 at epoch 3 to 0.1432 at epoch 4, and further to 0.1093 at epoch 5. However, at epoch 6, a sharp increase to 0.2735 was recorded. Following this spike, the loss began to decrease again, stabilizing at 0.0630 by epoch 7, and continuing to decrease to 0.0518, 0.0410, and 0.0389 in the subsequent epochs. This anomaly was likely due to the changes in the model's parameters during training, but it did not significantly affect the overall convergence pattern. However, VGG-16 with the Adam optimizer (4th run) did not exhibit this unusual behavior.

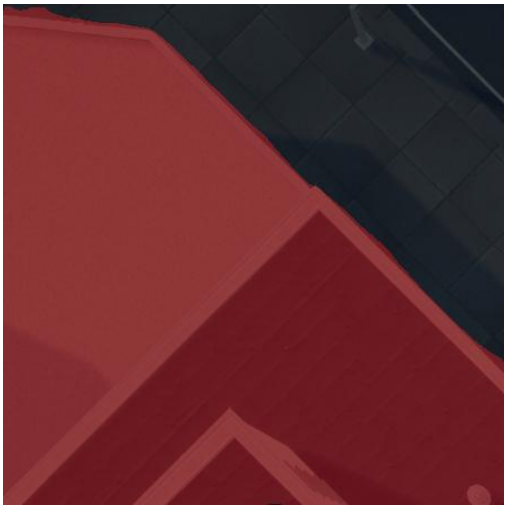
Run	Backbone	Epochs	BS	Opt	lr	IoU	P	R
1	ResNet-50	10	8	SGD	1e-2	98.12%	98.83%	99.27%
2	ResNet-50	10	8	Adam	1e-2	98.41%	98.98%	99.42%
3	VGG-16	10	8	SGD	1e-2	97.94%	98.90%	99.01%
4	VGG-16	10	8	Adam	1e-2	98.85%	99.21%	99.64%

Table 2.3: U-Net Training Results

Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate, R = Recall, P = Precision



(a) Input Image.



(b) Predicted mask from U-Net.

Mask R-CNN

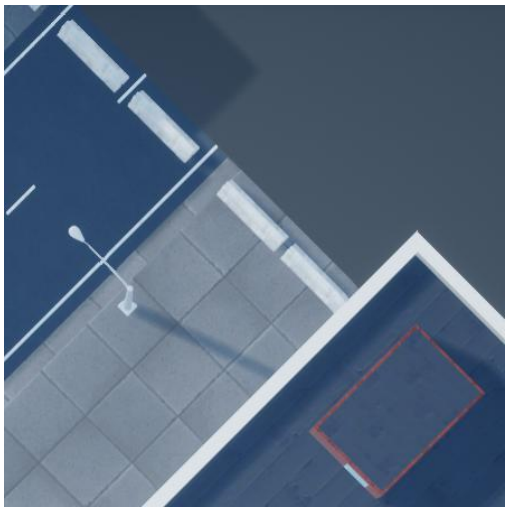
Table 2.4 summarizes the results of training with ResNet backbones, comparing different configurations of epochs, optimizers, and learning rates. The training runs for both ResNet-50 and ResNet-101 have been conducted for 10 epochs with a batch size of 8 and a learning rate of 1e-2. When using the SGD optimizer, both ResNet-50 and ResNet-101 have performed similarly. On the other hand, when Adam has been used as the optimizer, both ResNet-50 and ResNet-101 have also performed similarly, but ResNet-101 has achieved slightly higher precision and recall values. However, the IoU and overall performance with the Adam optimizer have dropped slightly in comparison to the SGD optimizer.
These findings suggest that the choice of optimizer has played a significant role in model performance, with SGD generally yielding higher results. Therefore, training configuration choices, such as optimizer and backbone selection, should be

carefully considered depending on the specific requirements of the task, such as computational efficiency or detection accuracy.

Run	Backbone	Epochs	BS	Opt.	lr	IoU	P	R
1	ResNet-50	10	8	SGD	1e-2	97.98%	95.1%	96.3%
2	ResNet-101	10	8	SGD	1e-2	98.09%	94.98%	96.2%
3	ResNet-50	10	8	Adam	1e-2	96.48%	88.9%	90.6%
4	ResNet-101	10	8	Adam	1e-2	96.73%	90.2%	91.9%

Table 2.4: Mask R-CNN Training Results

Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate, R = Recall, P = Precision



(a) Input Image.

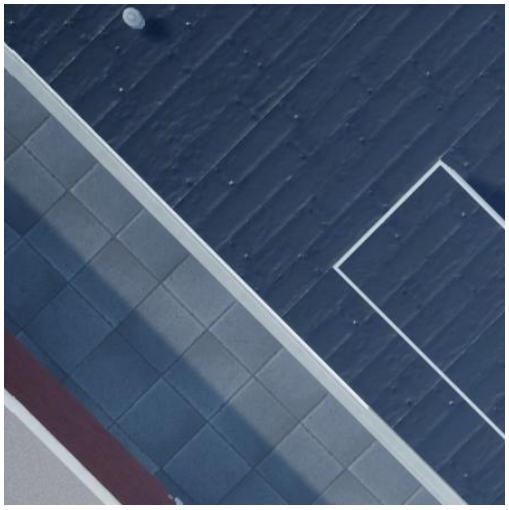
(b) Predicted mask from Mask R-CNN.

YOLOv8

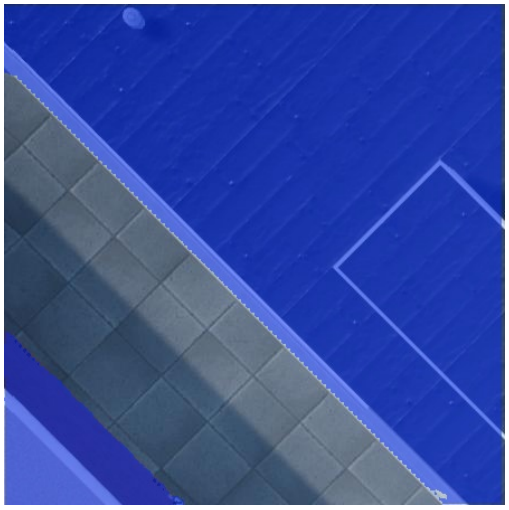
Run	Backbone	Epochs	BS	Opt	lr	IoU	P	R
1	ResNet-50	20	8	SGD	1e-2	95.81%	97.2%	84.5%
2	ResNet-50	20	8	Adam	1e-2	95.70%	93.7%	81.7%

Table 2.5: YOLOv8 Training Results

Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate, R = Recall, P = Precision



(a) Input Image.



(b) Predicted mask from YOLOv8.

2.4.5 Facility Synthetic Map

Due to the RAM limitations encountered during the urban map simulation, the facility map simulation has been conducted in two phases, each covering half of the map to avoid GPU constraints. Both phases have been completed successfully on the first attempt, with no issues encountered. Following this, the ground truth labels have been modified according to the number of classes used in each training.

The parameters used in the matlab script to control the simulation of the facility synthetic map were as follows:

Drone Height	100 m
Sweep Angle	45°
Image Width and Height	1080x1080
Focal length	35 mm
GSD	2.114 cm/pixel
UAV SampleTime	0.1 s
Nth Frame Interval	20
Look Ahead Distance	5 cm/s
WpSamplingFactor	8 m

Table 2.6: Simulation Parameters.

For the facility map dataset, training has been conducted exclusively with DeepLabv3+ 2.8 and U-Net 2.7 to explore the performance of alternative backbones, as the main results have been generated using only the ResNet-50 backbone.

Table 2.7 shows how different training settings, such as the choice of optimizer, number of epochs, and learning rate have affected U-Net performance. The results indicate that both the choice of optimizer and backbone significantly influence performance across IoU, Precision, and Recall metrics for gable and flat rooftop classifications.

With the VGG-16 backbone, the use of the Adam optimizer in run 2 leads to a notable improvement across all metrics compared to SGD at the same number of epochs and learning rate, suggesting that Adam facilitates faster and more effective convergence. For example, at 10 epochs with a learning rate of 1e-2, Adam achieves significantly higher IoU, Precision, and Recall scores than SGD. Extending training duration to 20 epochs with SGD (run 4) also increases performance metrics but still does not match the results obtained with Adam at just 10 epochs, highlighting Adam's efficiency in achieving higher accuracy with shorter training times. Furthermore, changing the learning rate from 1e-2 to 1e-4 in VGG-16 (run 5) while using SGD results in a slight decrease in performance, suggesting that the higher learning rate is beneficial when using this configuration.

Run	Backbone	Epochs	BS	Opt	lr	IoU		Precision		Recall	
						Gable	Flat	Gable	Flat	Gable	Flat
1	VGG-16	10	8	SGD	1e-2	94.22%	88.39%	97.9%	94.06%	96.17%	93.61%
2	VGG-16	10	8	Adam	1e-2	99.04%	95.78%	99.62%	98.88%	99.42%	96.83%
3	VGG-16	15	8	SGD	1e-2	96.93%	92.12%	98.91%	96.27%	97.98%	95.53%
4	VGG-16	20	8	SGD	1e-2	98.17%	94.62%	99.57%	97.59%	98.59%	96.89%
5	VGG-16	20	8	SGD	1e-4	96.83%	91.96%	99.17%	96.71%	97.62%	94.92%

Table 2.7: U-Net Training Results with 2 classes.

Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate

Table 2.8 summarizes all training conducted with DeepLabv3+ across different configurations, including variations in the number of epochs, and optimizers. Increasing the training duration from 10 to 20 epochs with the Xception backbone has notably enhanced the model's ability to generalize across both roof categories, suggesting that extended training has allowed more effective feature learning. Furthermore, the transition from the SGD optimizer to Adam has generally led to further performance improvements, underscoring Adam's adaptability in updating weights effectively and refining the model's detection capabilities. Comparing the two backbones, Xception and MobileNetV2, reveals that both perform well, but Xception has shown slightly better results with Adam, suggesting a stronger fit between this backbone and the optimizer. MobileNetV2 has still performed effectively and may offer practical advantages due to its lower computational and memory demands. Fine-tuning these configurations has helped make the models more capable of accurately detecting roof types, showing that both training and architecture decisions have a clear impact on the model's overall performance.

Run	Backbone	Epochs	BS	Opt	lr	IoU		Precision		Recall	
						Gable	Flat	Gable	Flat	Gable	Flat
1	Xception	10	8	SGD	1e-2	95.42%	89.17%	98.2%	93.93%	97.12%	94.62%
2	Xception	20	8	SGD	1e-2	98.59%	95.34%	99.33%	98.27%	99.25%	96.97%
3	Xception	20	8	Adam	1e-2	99.38%	97.07%	99.66%	99.09%	99.72%	97.94%
4	MobileNetV2	20	8	SGD	1e-2	98.9%	95.78%	99.08%	97.86%	99.81%	97.84%
5	MobileNetV2	20	8	Adam	1e-2	99.24%	96.88%	99.57%	98.93%	99.66%	97.91%

Table 2.8: DeepLabv3+ Training Results with 2 classes.

Note: BS = Batch Size, Opt = Optimizer, lr = learning Rate, R = Recall, P = Precision

Chapter 3

Results

This chapter presents the results of the final phase of this work for both maps. As mentioned earlier, the final phase involved training all four networks using the same dataset and hyperparameters, which are summarized in the table below:

Hyperparameter	**Value**
Backbone	ResNet-50
Epochs	20
Batch size	8
Learning rate	0.01
Optimizer	SGD & Adam
Momentum	0.9
Weight decay	0.0001
Automatic Mixed Precision (AMP)	True

Table 3.1: Final Phase Hyperparameters

The evaluation of all networks was conducted using three primary metrics: Intersection over Union (IoU), Precision, and Recall. These metrics are crucial for assessing each model's effectiveness in accurately detecting rooftop areas within the segmentation task. An IoU threshold of 0.5 was set for the predictions.

3.1 Urban Synthetic Map

Table 3.2 presents the results achieved by the four CNN models trained using the Stochastic Gradient Descent (SGD) optimizer.

Across all metrics, U-Net demonstrated the highest performance, achieving the highest IoU at 98.6%, Precision at 99.14%, and Recall at 99.45%. This suggests that U-Net was most effective in accurately delineating rooftop regions, as it is shown in Figure 3.1b. DeepLabv3+ also showed robust performance, closely following U-Net in accuracy. It achieved an IoU of 98.44%, Precision of 99.05%, and Recall of 99.38%. The similarity of DeepLabv3+'s metrics to those of U-Net indicates that it was nearly as effective in identifying rooftops, with only slight differences in prediction detail seen in Figure 3.1a. Mask R-CNN, however, exhibited lower performance relative to U-Net and DeepLabv3+, it was less precise and comprehensive in identifying rooftops, as shown by its predicted mask, Figure 3.1c, which displayed some gaps in rooftop segmentation. YOLOv8's performance, while effective, was more variable. Although it achieved a high Precision of 97.2% and a satisfactory IoU of 95.81%, its Recall dropped significantly to 84.5%. This indicates that YOLOv8, as observed in its predicted mask, Figure 3.1d, may have missed certain rooftop areas, leading to lower overall completeness in detection compared to the other models.

Networks / Metrics	DeepLabv3+	U-Net	Mask R-CNN	YOLOv8
IoU	98.44%	98.6%	96.13%	95.81%
Precision	99.05%	99.14%	89%	97.20%
Recall	99.38%	99.45%	90.6%	84.50%

Table 3.2: Performance evaluation with 'SGD'.

Table 3.3 illustrates the performance of the four models trained using Adam optimizer.

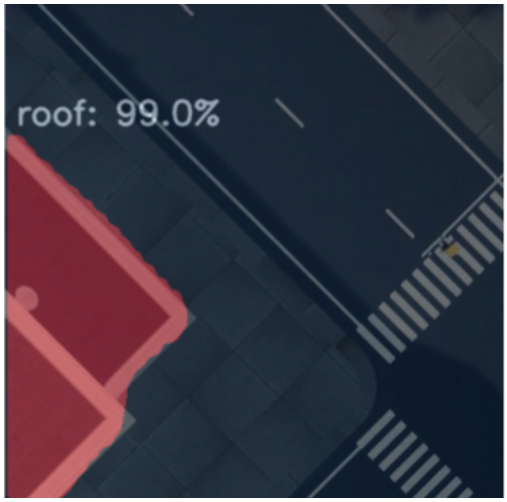
The table illustrates that U-Net and DeepLabv3+ demonstrate superior performance across all metrics, particularly excelling in IoU (Intersection over Union), Precision, and Recall compared to Mask R-CNN and YOLOv8. Both models achieve nearly identical IoU scores, indicating that they have highly similar capabilities for accurately segmenting regions of interest, with U-Net marginally outperforming DeepLabv3+ by a small fraction.

In terms of Precision, U-Net slightly edges out DeepLabv3+, reflecting its ability to more accurately identify true positives with fewer false positives. This high Precision is advantageous for applications where accuracy in identification is critical. Recall values are very close between U-Net and DeepLabv3+, showing that both models are highly capable of identifying relevant instances without missing too many.

Mask R-CNN shows a moderate decline in both IoU and Precision compared

(a) Predicted mask from DeepLabv3+.

(b) Predicted mask from U-Net.



(c) Predicted mask from Mask R-CNN.

(d) Predicted mask from YOLOv8.

Figure 3.1: Predicted Segmentation Masks Across Models.

to the top-performing models, suggesting that it may be less accurate in identifying boundaries or relevant objects. However, it retains a relatively high Recall, indicating that it can capture most relevant instances but may produce more false positives, reducing its Precision.

YOLOv8, on the other hand, has the lowest metrics across the board. Its lower IoU and Recall suggest that it is less reliable in accurately capturing regions of interest, likely due to its design prioritizing speed over segmentation precision. Additionally, its lower Precision reflects a higher rate of false positives, making it less ideal for tasks requiring high accuracy in pixel-level segmentation.

Overall, U-Net and DeepLabv3+ emerge as the top choices for applications needing high segmentation accuracy, with U-Net having a slight advantage in Precision and IoU. Mask R-CNN performs well but lags behind in exact precision, while YOLOv8 shows limitations in handling precise segmentation tasks, making it less suitable when accuracy is prioritized over processing speed.

Metrics \ Networks	DeepLabv3+	U-Net	Mask R-CNN	YOLOv8
IoU	98.77%	98.81%	97.91%	95.70%
Precision	99.19%	99.25%	93.4%	81.70%
Recall	99.57%	99.55%	94.6%	93.70%

Table 3.3: Performance evaluation with 'Adam'.

3.2 Facility Map

3.2.1 Synthetic Images

For this map, two datasets have been created: one comprising synthetic images and the other real images. The synthetic images have been generated through simulation using an aerial image of the entire facility, captured by a UAV and processed in Unreal Engine software, as previously explained in Section 2.4.1. Initially, training has been conducted on the synthetic dataset, with all roof areas considered as a single class, "roof". The best-performing checkpoint from each model has been saved and applied to real images to test the model's general ability to detect rooftops without differentiating roof types. A second training phase has been conducted, this time classifying roofs as "flat" and "gable". The best checkpoint from this phase has similarly been used to assess each model's performance in recognizing and classifying the two roof types on real images. This comparison between synthetic and real datasets has provided insights into the models' generalization abilities and effectiveness for real-world applications.

Table 3.4 compares the performance of the evaluated models using SGD optimizer. Among the models, DeepLabv3+ demonstrates superior performance across all metrics, with an IoU of 97.86%, Precision of 99.06%, and Recall of 98.77%. These results highlight its capability to accurately classify rooftop pixels while minimizing false positives and negatives. U-Net, with an IoU of 95.36%, Precision of 97.83%, and Recall of 97.42%, performs similarly to DeepLabv3+, showcasing its effectiveness in segmentation tasks. Mask R-CNN and YOLOv8, while still performing well, achieve comparatively lower scores. Mask R-CNN records an IoU of 93.61%, Precision of 85.8%, and Recall of 87.7%, reflecting its strengths in instance segmentation but

also its limitations in capturing fine-grained details. YOLOv8, primarily an object detection model, achieves an IoU of 89.02%, Precision of 93.7%, and Recall of 80.8%. The lower recall score indicates that it tends to miss some rooftop regions, likely due to its architectural emphasis on bounding box detection rather than pixel-level accuracy.

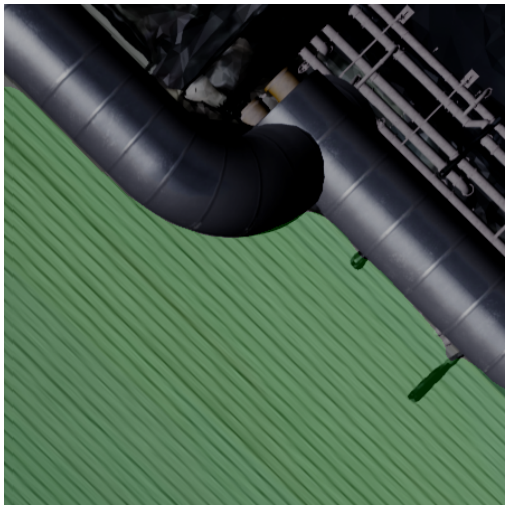
Overall, DeepLabv3+ emerges as the most effective model for rooftop detection and classification, closely followed by U-Net. Both models excel in tasks requiring high segmentation accuracy. In contrast, Mask R-CNN and YOLOv8, while proficient in their respective domains, show limitations in achieving the same level of precision and recall, particularly in the nuanced requirements of rooftop analysis.

Metrics \ Networks	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
IoU	97.86%	95.36%	93.61%	89.02%
Precision	99.06%	97.83%	85.8%	93.7%
Recall	98.77%	97.42%	87.7%	80.8%

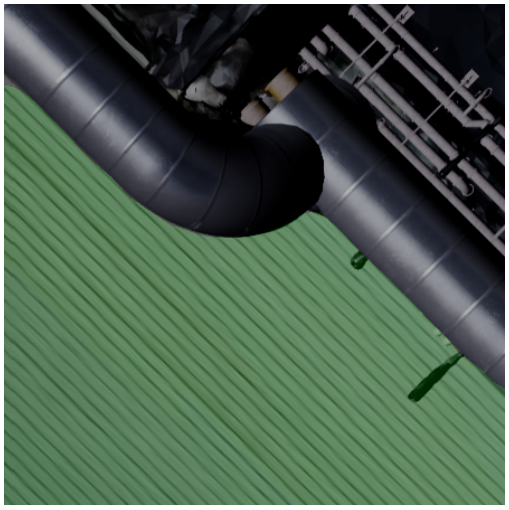
Table 3.4: Performance evaluation with SGD on 1 class.

Table 3.5 provides a comprehensive comparison of the four models' performance when trained with the Adam optimizer. Among the models, U-Net emerges as the top performer, achieving the highest IoU (98.97%), Precision (99.45%), and Recall (99.51%). These results highlight U-Net's superior ability to accurately segment and classify rooftop structures, excelling in both precision and completeness of detection. DeepLabv3+ also demonstrates robust performance, achieving an IoU of 97.9%, Precision of 99.08%, and Recall of 98.8%. While slightly behind U-Net, these metrics underscore its consistent strength in semantic segmentation tasks, particularly in scenarios requiring high precision and reliable segmentation boundaries. Mask R-CNN, with an IoU of 94.14%, Precision of 86.8%, and Recall of 87.2%, performs moderately well but falls behind the pixel-level segmentation-focused models. Its architecture, optimized for instance segmentation, likely contributes to this discrepancy, as its strength lies in delineating object boundaries rather than holistic segmentation accuracy. YOLOv8, achieving an IoU of 88.94%, Precision of 92.1%, and Recall of 76.3%, shows notable proficiency in precision but struggles with recall. This reflects YOLOv8's focus on object detection, which, while effective for identifying objects, is less suited for pixel-wise classification tasks.

In conclusion, U-Net is the most effective model for rooftop detection and classification in this evaluation, followed closely by DeepLabv3+. Mask R-CNN and YOLOv8, while competent, are less suitable for pixel-level segmentation tasks, with YOLOv8 particularly limited by its lower recall metric.

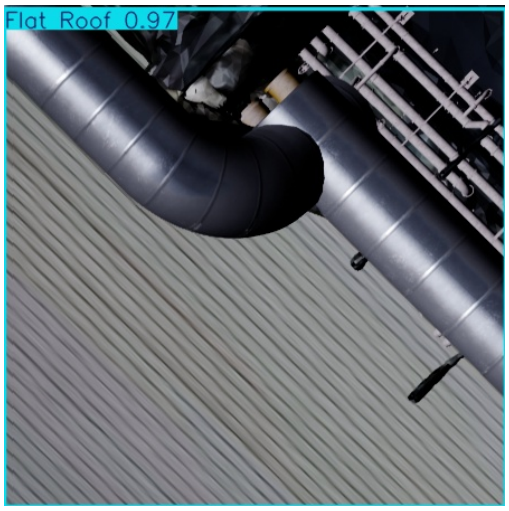


(a) Predicted mask from DeepLabv3+.



(b) Predicted mask from U-Net.

(c) Predicted mask from Mask R-CNN.



(d) Predicted mask from YOLOv8.

Metrics \ Networks	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
IoU	97.9%	98.97%	94.14%	88.94%
Precision	99.08%	99.45%	86.8%	92.1%
Recall	98.8%	99.51%	87.2%	76.3%

Table 3.5: Performance evaluation with Adam on 1 class.

Table 3.6 compares the performance of the four networks in detecting and

classifying gable and flat rooftops using SGD optimizer.

For gable roofs, U-Net achieved the highest IoU (99.03%) and Precision (99.62%), closely followed by DeepLabv3+ with strong IoU and Precision values (98.3% and 99.51%, respectively). Both models demonstrated high accuracy in identifying gable rooftops, with U-Net slightly leading in precise detection. Mask R-CNN, while achieving a reasonable IoU (97.61%), had significantly lower Precision (92.43%) and Recall (86.62%), indicating some variability in distinguishing gable roof instances. YOLOv8 showed lower performance on gable roofs, particularly in Recall (43.5%), suggesting a challenge in accurately detecting these structures.

For flat roofs, U-Net again led in IoU (96.98%) and Recall (98.16%), with DeepLabv3+ close behind. These results suggest that both networks are highly effective for flat roof segmentation, with U-Net consistently offering a slight edge in precision across roof types. Mask R-CNN achieved a comparable IoU for flat roofs (96.62%) but exhibited reduced Precision (82.03%), suggesting a tendency toward over-segmentation. YOLOv8's IoU (85.11%) and Recall (82.7%) for flat roofs were its best scores but remained lower than those of the other models, highlighting a trade-off between speed and detection precision in YOLOv8's performance.

In summary, U-Net and DeepLabv3+ have shown overall robustness across both rooftop types, with U-Net slightly outperforming in precision and recall for both gable and flat roofs. Mask R-CNN, while capable, appears to encounter challenges with detailed segmentation for gable roofs, potentially due to instance-based constraints. YOLOv8, optimized for rapid object detection, presents a trade-off with lower segmentation accuracy, making it less suitable for more complex structures like gable roofs. This analysis underscores the balance between segmentation accuracy and model design, where U-Net and DeepLabv3+ achieve strong performance at the cost of computational complexity, while YOLOv8 remains a faster but less precise option.

Roof Name	Networks / Metrics	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
Gable Roof	IoU	95.3%	99.03%	97.61%	92.01%
	Precision	97.51%	99.62%	92.43%	79.8%
	Recall	96.77%	99.41%	86.62%	43.5%
Flat Roof	IoU	95.04%	96.98%	96.62%	85.11%
	Precision	97.9%	98.77%	82.03%	83%
	Recall	97.02%	98.16%	86.63%	82.7%

Table 3.6: Performance Evaluation with 'SGD' on 2 Classes.

When trained with Adam optimizer, table 3.7, the U-Net model has achieved the highest performance among all networks. For both gable and flat roofs, U-Net has produced the top IoU scores, reaching 99.15% for gable roofs and 97.04% for

flat roofs. These high IoU values reflect U-Net's precise segmentation capability and its adaptability to different rooftop types. DeepLabv3+ has also performed well, achieving IoU, Precision, and recall scores close to those of U-Net, indicating it consistently captured roof details without significant false positives or missed detections. Mask R-CNN has shown balanced performance but lower than U-Net and DeepLabv3+. For gable roofs, it reached 97.59% IoU and demonstrated improved recall (95.13%), suggesting that it can detect more instances, although with some compromises in boundary accuracy, as indicated by a lower precision for gable roofs (86.41%). For flat roofs, Mask R-CNN's precision (87.85%) and recall (94.83%) were robust, underscoring its reliability in detecting simpler shapes. YOLOv8, while achieving reasonable accuracy, has shown the lowest metrics across the board, particularly for gable roofs where recall is limited to 32.3%, indicating fewer true detections of this complex shape. However, YOLOv8's precision has been stronger for flat roofs (93%), suggesting better specificity in detecting and classifying simpler structures without unnecessary false positives. YOLOv8 for gable roofs exhibited a high IoU and precision but a very low recall. This indicates that while the model accurately predicted the rooftops it detected, it failed to identify a significant number of them, leading to false negatives. This imbalance could be due to YOLOv8 being more selective about the bounding boxes it accepts as correct, focusing on high confidence predictions, which can result in missed detections. The relatively low recall in this case suggests that while the model's precision is high, it doesn't detect all rooftops, possibly missing smaller or less clear examples.

Overall, U-Net has achieved the best balance of precision and recall, especially for complex roof types, followed closely by DeepLabv3+. Mask R-CNN performs well, especially in recall for simpler flat roofs, while YOLOv8 shows a tendency to perform more effectively with simpler structures, such as flat roofs, where it can avoid false positives with good precision.

Roof Name	Networks / Metrics	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
Gable Roof	IoU	95.69%	99.15%	97.59%	91.65%
	Precision	98.24%	99.71%	86.41%	85.7%
	Recall	97.36%	99.44%	95.13%	32.3%
Flat Roof	IoU	96.06%	97.04%	96.92%	84.84%
	Precision	89.46%	98.79%	87.85%	93%
	Recall	94.64%	98.21%	94.83%	69.8%

Table 3.7: Performance Evaluation with 'Adam' on 2 Classes.

3.2.2 Real Images

The best checkpoint from the training on the synthetic dataset for each model was saved and used to evaluate the model's performance on the real images. This approach ensured that the models leveraged their best capability, as captured during the synthetic image training, to assess how well they generalized to real-world scenarios. However, despite using the best-performing checkpoint, the models still encountered challenges due to the inherent differences between synthetic and real images, such as variations in lighting, texture, and environmental factors. These challenges contributed to the noticeable decrease in performance when transitioning from synthetic to real image datasets.

Table 3.8 highlights the performance of the four models when applied to real images. In this scenario, DeepLabv3+ achieves the best overall performance, with an IoU of 88.88% and a Recall of 95.16%, reflecting its ability to accurately identify and segment rooftop structures. Its Precision of 93.08% further emphasizes its balance between detecting true positives and avoiding false positives, making it well-suited for real-world applications. U-Net, while slightly behind DeepLabv3+ in IoU (85.51%), excels in Precision (95.21%), surpassing all other models in this metric. This indicates U-Net's strong capability in correctly identifying rooftop pixels while minimizing false positives. However, its Recall (89.36%) lags behind that of DeepLabv3+, suggesting a relative limitation in capturing all relevant rooftop regions. Mask R-CNN shows moderate performance, with an IoU of 75.9%, Precision of 72.5%, and Recall of 65.2%. These results are indicative of its focus on instance segmentation, which may not be as effective for large-scale or complex segmentation tasks as pixel-based models like DeepLabv3+ and U-Net. YOLOv8, with an IoU of 73.11%, Precision of 84.6%, and Recall of 60.8%, struggles to match the segmentation-focused models. While its Precision is relatively high, its Recall is the lowest among all models, reflecting challenges in identifying all instances of rooftops in real-world images. YOLOv8's object detection focus limits its effectiveness in dense or cluttered segmentation scenarios.

In summary, DeepLabv3+ and U-Net demonstrate strong adaptability to real-image datasets, with DeepLabv3+ slightly outperforming U-Net in IoU and Recall, while U-Net achieves the highest Precision. Mask R-CNN and YOLOv8, while useful for their respective strengths, are less effective for pixel-level segmentation of complex real-world rooftop imagery.

Networks / Metrics	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
IoU	88.88%	85.51%	75.9%	73.11%
Precision	93.08%	95.21%	72.5%	84.6%
Recall	95.16%	89.36%	65.2%	60.8%

Table 3.8: Performance evaluation of real images with SGD on 1 class.

Table 3.9 presents the performance of the four models on real-world rooftop images when using the Adam optimizer. DeepLabv3+ continues to demonstrate a strong overall performance with an IoU of 85.3%, the highest among the models, indicating its robustness in accurately segmenting rooftops. Its Precision of 92.43% and Recall of 91.7% reflect a balanced ability to correctly identify rooftops while minimizing false positives and false negatives. U-Net closely follows DeepLabv3+, with a slightly lower IoU of 85.06%. However, its Recall of 94.97% surpasses that of DeepLabv3+, showing that U-Net is highly effective at capturing all relevant rooftop regions. Nevertheless, its Precision of 89.07% suggests a slight tendency to include more false positives compared to DeepLabv3+. Mask R-CNN, with an IoU of 75.7%, falls behind the segmentation-focused models but shows consistent performance compared to its results with the SGD optimizer. Its Precision (70.4%) and Recall (64.2%) reflect its limitation in handling complex real-world rooftop segmentation tasks, which require a finer level of detail. YOLOv8 achieves an IoU of 75.43%, comparable to Mask R-CNN, but its Recall (59.3%) remains the lowest among all models. While its Precision of 82.8% indicates a reasonable ability to avoid false positives, the low Recall underscores its challenge in identifying all rooftop instances in real-world images.

Overall, DeepLabv3+ and U-Net continue to stand out as the most reliable models for real-world rooftop detection and segmentation tasks, with DeepLabv3+ providing balanced performance and U-Net excelling in Recall. In contrast, Mask R-CNN and YOLOv8, while serviceable, are less competitive in pixel-level segmentation for this application.

Networks / Metrics	DeepLabv3+	U-Net	Mask RCNN	YOLOv8
IoU	85.3%	85.06%	75.7%	75.43%
Precision	92.43%	89.07%	71.8%	82.8%
Recall	91.7%	94.97%	68.3%	59.3%

Table 3.9: Performance evaluation of real images with Adam on 1 class.

Table 3.10 presents the performance evaluation of the four models trained on real images using the SGD optimizer.

When examining the models' abilities to detect and classify the Gable Roof, U-Net emerges as the most balanced model, consistently performing well across different metrics. It achieves the highest recall and a very competitive IoU score, reflecting its ability to detect both the foreground (roof) and background areas with relatively good precision. This suggests that U-Net is effective in distinguishing the roof structure from its surroundings, especially in cases with complex shapes or boundaries. DeepLabv3+ follows closely behind U-Net in terms of performance, demonstrating strong precision and a respectable IoU score. Its slightly lower recall compared to U-Net indicates that it might miss some true positives (i.e., certain regions of the roof), leading to a somewhat higher false-negative rate. Nonetheless, DeepLabv3+ remains highly capable for roof detection tasks, particularly in terms of precision. Mask R-CNN and YOLOv8, on the other hand, face notable challenges in detecting and classifying Gable Roofs. Although Mask R-CNN shows a solid precision score, its relatively lower recall and IoU suggest that it struggles to detect the full extent of the roof area, likely due to its tendency to miss smaller or less distinct features of the Gable Roof. YOLOv8 demonstrates the most significant issues, with poor recall and precision scores. This indicates that YOLOv8 struggles not only with missing roof regions (false negatives) but also with accurately identifying the roof when it is present (false positives), making it less reliable for this particular task.

For the Flat Roof, the models' performances are more competitive. DeepLabv3+ leads in terms of both IoU and precision, reflecting its overall robustness in detecting the roof structure accurately. U-Net is close behind, showing excellent recall but slightly lower precision compared to DeepLabv3+, indicating that it might detect more roof regions but at the cost of introducing some false positives. Mask R-CNN, while performing decently in terms of recall, lags behind in both precision and IoU, which suggests that it is less effective in classifying and localizing the Flat Roof compared to the top performers. YOLOv8 demonstrates better recall than Mask R-CNN, suggesting that it can identify a broader range of roof areas, but its very low precision means that many of these detections are false positives. This makes YOLOv8 the least effective model for accurately detecting and classifying the Flat Roof, despite its higher recall.

Overall, U-Net and DeepLabv3+ stand out as the more reliable models for both roof types, with U-Net excelling in recall and DeepLabv3+ leading in precision. Mask R-CNN and YOLOv8 both face challenges in terms of precision and recall, with YOLOv8 performing particularly poorly in Gable Roof detection.

Roof Name	Networks / Metrics	DeepLabv3+	U-Net	Mask R-CNN	YOLOv8
Gable Roof	IoU	60.45%	61.86%	52.54%	44.09%
	Precision	90.49%	90.07%	62.78%	53.6%
	Recall	64.55%	66.38%	60.1%	15.6%
Flat Roof	IoU	85.18%	84.91%	76.99%	73.98%
	Precision	85.89%	85.57%	67.78%	38%
	Recall	69.04%	69.1%	70.26%	72.2%

Table 3.10: Performance Evaluation of Real Images with 'SGD'.

Table 3.11 presents the performance of the four models trained on real images using Adam optimizer. For the Gable Roof, U-Net performs the best overall with an IoU of 58.32%, slightly outperforming DeepLabv3+ (56.05%). The relatively higher Recall (60.48%) of U-Net indicates that it detects a good portion of the true positive regions of the roof, though not as many as DeepLabv3+ which has a higher Recall of 66.62%. Despite U-Net's slightly higher Recall, its Precision (94.22%) is the highest, showing that it tends to produce fewer false positives, though this could be at the expense of detecting some of the true positives, as the relatively low Recall suggests. DeepLabv3+ has a well-balanced performance with high Recall and Precision. On the other hand, Mask R-CNN and YOLOv8 show relatively poorer performance. Mask R-CNN has the lowest IoU at 51.07%, which reflects its weaker segmentation ability for the Gable Roof. It also shows a Precision of 57.45% and Recall of 60.33%, suggesting that it struggles more with false positives than the other models. YOLOv8, while achieving a reasonable IoU of 53.5%, has a significantly low Recall of 24.1%. This low Recall indicates that YOLOv8 is missing a large proportion of the true positives, even though its Precision (45.2%) is relatively better compared to Mask R-CNN.

In the Flat Roof category, the performance improves across all models. U-Net again leads the way with an IoU of 79.94% and a solid Precision of 80.37%, showing its balanced performance in detecting and correctly segmenting flat roofs. It also performs well with a Recall of 80.37%, indicating that it does not miss many true positives and maintains low false negatives. DeepLabv3+ follows closely with an IoU of 79.83%, showing that its segmentation capability is almost on par with U-Net, but its Precision of 85.99% suggests it is slightly more prone to false positives compared to U-Net. However, it achieves a slightly lower Recall (71.77%), indicating that it misses more true positives than U-Net, though the difference isn't drastic. Mask R-CNN and YOLOv8 show lower IoU scores (70.59% and 62.42%, respectively) and slightly poorer performance in Precision. YOLOv8 demonstrates a lower Precision (62.3%) and a higher Recall (77.8%), suggesting that it is better at detecting more true positives but at the cost of generating more

false positives. Mask R-CNN has a Precision of 62.78%, with Recall (65.22%) also showing a moderate balance, although it still underperforms compared to U-Net and DeepLabv3+.

In summary, U-Net consistently outperforms the other models in both IoU and Recall, particularly in the Flat Roof category, where it achieves a perfect balance between detecting true positives and avoiding false positives. DeepLabv3+ shows strong Recall but sacrifices some Precision, particularly for the Gable Roof. Mask R-CNN and YOLOv8 tend to struggle in terms of segmentation accuracy, with YOLOv8 showing a significant weakness in Recall on both roof types, missing many true positive regions. Therefore, U-Net emerges as the most reliable model for roof segmentation tasks, especially when both high Precision and Recall are required. DeepLabv3+ is also a good choice, especially when precise segmentation is needed, while YOLOv8 and Mask R-CNN are less suited for this task due to their relatively lower segmentation performance and high false negatives (in the case of YOLOv8).

Roof Name	Networks / Metrics	DeepLabv3+	U-Net	Mask R-CNN	YOLOv8
Gable Roof	IoU	56.05%	58.32%	51.07%	53.5%
	Precision	90.94%	94.22%	57.45%	45.2%
	Recall	66.62%	60.48%	60.33%	24.1%
Flat Roof	IoU	79.83%	79.94%	70.59%	62.42%
	Precision	85.99%	80.37%	62.78%	62.3%
	Recall	71.77%	80.37%	65.22%	77.8%

Table 3.11: Performance Evaluation of Real Images with 'Adam'.

Chapter 4

Conclusion

This study has focused on detecting and classifying rooftops of industrial facilities, specifically distinguishing between flat and gable roofs, using aerial images captured via UAVs. Four state-of-the-art deep learning models: DeepLabv3+, U-Net, Mask R-CNN, and YOLOv8 have been employed to perform semantic segmentation and object detection tasks. The models have been trained and validated using synthetic datasets and subsequently tested on real-world images to evaluate their robustness and generalizability. This dual approach has provided valuable insights into the performance and challenges of transitioning from synthetic to real-world scenarios.

The experimental results have revealed that synthetic datasets serve as an effective tool for training models due to their consistent and controlled nature, which has led to high performance metrics across all models. When considering a single class (roofs), the models have demonstrated excellent results on the synthetic dataset, achieving IoU scores exceeding 95% for segmentation-based models like DeepLabv3+ and U-Net, and robust performance from detection-based models like Mask R-CNN and YOLOv8. However, when tested on real images, the performance has declined significantly. The IoU scores have dropped by approximately 10-15% for most models, with YOLOv8 showing the largest drop. This gap has highlighted the challenge of domain adaptation, as synthetic data lacks the variability and complexities present in real-world imagery, such as lighting conditions, occlusions, and structural irregularities.

The performance discrepancy has become even more pronounced when transitioning to a two-class problem (flat and gable roofs). In the synthetic domain, U-Net has consistently achieved the highest accuracy, with IoU scores above 95% for both roof types, while other models have also performed competitively. On real-world images, however, the IoU scores have dropped dramatically, particularly for gable roofs, where models like YOLOv8 have struggled significantly. DeepLabv3+ and U-Net have exhibited better generalization compared to detection-based models, maintaining higher IoU and recall values, yet all models have faced challenges

with false negatives, especially for gable roofs. These observations underscore the complexity of real-world datasets, where factors like class imbalance, intricate roof geometries, and UAV image distortions have further challenged model performance. Mask R-CNN and YOLOv8 showed a more significant decline, struggling with the real-world complexity due to their focus on object detection. YOLOv8, while designed for real-time object detection and fast processing, demonstrated lower precision in the real-world scenario, which made it less desirable for this task. Although YOLOv8 excels at speed, it is primarily designed for applications where real-time performance is critical, rather than high precision. Since precision is a key factor in this work, YOLOv8's emphasis on speed over accuracy makes it less suitable for detecting and classifying rooftops, where detailed segmentation and accuracy are paramount.

In summary, this research has demonstrated the potential of deep learning models for industrial rooftop detection and classification but emphasizes the importance of real-world validation. While synthetic datasets have provided an essential foundation for training, their limitations in representing real-world variability highlight the need for domain adaptation techniques or real-world data augmentation.

4.1 Future Work

Despite the significant advancements achieved in this study, several limitations have been identified, which provide opportunities for future improvements. One of the main challenges encountered was the imbalance in the dataset, particularly the lower number of images containing gable roofs compared to flat roofs. This imbalance has likely contributed to the reduced performance of the models, especially in real-world scenarios where gable roofs were underrepresented. Future efforts should focus on increasing the size and diversity of the dataset, with a particular emphasis on collecting additional images of gable roofs. This would not only help balance the dataset but also improve the models' ability to generalize across different roof types.

Another avenue for improvement involves leveraging advanced data augmentation techniques to bridge the gap between synthetic and real-world datasets. Augmentation strategies such as varying lighting conditions, adding realistic noise, or simulating occlusions could help mimic real-world complexities, enhancing the robustness of model training.

Appendix A

Computer Vision

A.1 Challenges of Computer Vision

Computer vision, despite its advances, faces several challenges that researchers and continue to address:[1]

- Hardware limitations and innovations: The computational complexity of processing high volumes of visual data and integrating computer vision with other technologies poses significant challenges for hardware systems.
- Variability in Lighting Conditions and Adverse Weather Conditions: Changes in lighting can dramatically affect the visibility and appearance of objects in images.
- Occlusions: Objects can be partially or fully blocked by other objects, making detection and recognition difficult.
- Scale Variation: Objects can appear in different sizes and distances, complicating detection.
- Background Clutter: Complex backgrounds can make it hard to distinguish and segment objects properly.
- Viewpoint Variation: Objects can appear different when viewed from different angles.
- Deformations: Flexible or soft objects can change shape, and it is challenging to maintain consistent detection and tracking.
- Limited Data and Annotation: Training advanced models requires large datasets with accurate labeling, which can be costly and time-consuming.

A.2 Computer Vision Benefits

Computer vision has emerged as a transformative technology, offering numerous benefits across various fields by enabling automated analysis and interpretation of visual data, the following outlines some of these benefits:[1]

- Automation of Visual Tasks: Computer vision automates tasks that require visual cognition, significantly speeding up processes and reducing human error.
- Enhanced Accuracy: In many applications, such as medical imaging analysis, CV can detect anomalies more accurately and consistently than human.
- Real-Time Processing: CV enables real-time processing and interpretation of visual data, crucial for applications like autonomous driving and security surveillance, where immediate response is essential.
- Cost Reduction: By automating routine and labor-intensive tasks, computer vision reduces the need for manual labor, thereby cutting operational costs over time.
- Enhanced Safety: In industrial environments, computer vision can monitor workplace safety, detect unsafe behaviors, and ensure compliance with safety protocols, reducing the risk of accidents.
- Improved User Experience: In retail and entertainment, computer vision enhances customer interaction through personalized recommendations and immersive experiences like augmented reality.
- Data Insights: By analyzing visual data, businesses can gain insights into consumer behavior, operational bottlenecks, and other critical metrics, aiding in informed decision-making.
- Accessibility: Computer vision enhances accessibility by helping to create assistive technologies for the visually impaired, such as real-time text-to-speech systems or navigation aids.
- Innovation: As a frontier technology, computer vision drives innovation in many fields, from developing advanced healthcare diagnostic tools to creating interactive gaming systems.

Appendix B

Architectures of CNNs

B.1 CNN models

These are some CNN architectures that haven't been used in this work.

1. LeNet: The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.[56]

2. AlexNet: Proposed by Krizhevsky et al. (2012), it has 60 million parameters and 650,000 neurons. It consists of five convolutional layers, followed by three connected layers. Each convolutional layer is followed by a rectified linear unit (ReLU) used to activate outputs of convolutional layers.[82]

3. GoogLeNet: Lin et al. (2014) introduced Network in-network (NIN), which consists of a stack of mlpconv layers. It replaces convolution filters with a general nonlinear function approximator. Another feature of NIN is that it uses global average pooling to replace fully connected layers. It averages each feature map and feeds the resulted vector directly to softmax layer.

4. Inception Nets: Szegedy et al. (2015) proposed a new CNN architecture called Inception v1. It manages to increase the depth and width of the network while keeping the computing budget constant. The inception layers are repeated multiple times and formed GoogLeNet, a 22-layer deep model. GoogLeNet utilizes two ideas in NIN: the 1×1 Convolution and global average pooling. Afterward, Szegedy et al. (2016) introduced a set of tricks to improve the efficacy of the original design of Inception v1. It points out that convolutions with larger filters tend to be disproportionally expensive in terms of computation. It suggests replacing filters with the size of 5×5 with two stacked 3×3 filters. This design calls Inception v2. The authors also

mentioned a batch normalization (BN) auxiliary, which used normalization within each mini-batch data to normalize the output to a normal distribution of N (0,1), reducing changes in the distribution of internal neurons. They refer to this design as Inception v3. Inspired by ResNet, Szegedy et al. (2017) introduced Inception v4 as a simplified version of Inception v3. They combined Inception architecture with residual connections and create a new architecture called Inception-ResNet[82]

5. MobileNets (v1): Howard et al. (2017) presented a class of efficient models called MobileNets (v1), which used two simple global hyperparameters that efficiently trade-off between latency and accuracy.[82]

6. MobileNetV3: Howard et al. (2019) made MobileNetV3 come in two versions, MobileNetv3-Small and MobileNetv3-Large, which have lower and higher computation and storage requirements. MobileNetV3 applies neural architecture search (NAS) and NetAdapt algorithm to improve the performance.[82]

7. DenseNet: Based on the observation that convolutional networks are more accurate and faster, Huang et al. (2017) introduced DenseNet that connects all layers directly with each other. Using the feature maps from all preceding layers as inputs, the DenseNet can create L(L+1)/2 connections rather than L connections of traditional convolution networks. As a result, it has four advantages: alleviating the vanishing-gradient problem, strengthening feature propagation, encouraging feature reuse, and reducing the number of parameters.[82]

8. EfficientNet: Tan and Le (2019) found that make balance of network depth, width, and resolution can lead to better performance, and thus introduced EfficientNet. Overall, EfficientNet-B7 achieves the state-of-the-art 84.4% top-1 and 97.1% top-5 accuracy on ImageNet, while being 8.4x smaller and 6.1x faster on the inference than GPipe.[82]

9. RegNet: Radosavovic et al. (2020) presented a new network design paradigm called RegNet that combines the advantages of manual design and NAS. The RegNet design space can work perfectly across a wide range of flop regimes. Under similar training settings and flops, the RegNet model outperforms the popular EfficientNet model, which is up to 5-times faster on the GPU. Although the precision of RegNet is not a great improvement compare with EfficientNet, it proposes new ideas in the direction of design network design spaces.[82]

Appendix C

Performance Metrics

The following metrics, while commonly used to evaluate the performance of CNN models, have not been employed in this work:

1. **Confusion Matrix**: It is used to define a classification alorithm's performance.[73]

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Positive	False Positive (FP)	True Negative (TN)

Table C.1: Confusion Matrix.

2. **Accuracy**: It is defined as the ratio of correctly classified/predicted samples to the total number of samples.[73]

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{C.1}$$

3. **F1-Score**: It combines precision and recall to provide a single value representing a classification model's overall performance. It is defined as the harmonic mean of precision and recall, computed as:

$$\text{F1-Score} = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity} \tag{C.2}$$

   A higher F1 score indicates a better balance between precision and recall, suggesting that the model is not overly skewed towards one metric at the expense of the other.[73]

Appendix D

COCO Dataset

The COCO (Common Objects in Context) dataset is a large-scale object detection, segmentation, and captioning dataset. It is designed to encourage research on a wide variety of object categories and is commonly used for benchmarking computer vision models. It contains 330K images, with 200K images having annotations for object detection, segmentation, and captioning tasks. The dataset comprises 80 object categories, including common objects like cars, bicycles, and animals, as well as more specific categories such as umbrellas, handbags, and sports equipment. Annotations include object bounding boxes, segmentation masks, and captions for each image. COCO provides standardized evaluation metrics like mean Average Precision (mAP) for object detection, and mean Average Recall (mAR) for segmentation tasks, making it suitable for comparing model performance.[83]

The COCO dataset is split into three subsets:[83]

- Train2017: This subset contains 118K images used for training.
- Val2017: This subset has 5K images used for validation purposes during model training.
- Test2017: This subset consists of 20K images used for testing and benchmarking the trained models. Ground truth annotations for this subset are not publicly available..

The COCO dataset is widely used for training and evaluating deep learning models in object detection (such as YOLO, Faster R-CNN, and SSD), instance segmentation (such as Mask R-CNN), and keypoint detection (such as OpenPose). The dataset's diverse set of object categories, large number of annotated images, and standardized evaluation metrics make it an essential resource for computer vision researchers and practitioners.

Bibliography

[1] Simplilearn. «The Power of Computer Vision in AI: Unlocking the Future!» In: *Simplilearn* (2024). URL: https://www.simplilearn.com/computer-vision-article (cit. on pp. 5, 10–12, 57, 58).

[2] Junshan Liu, Michael Jennesse Jr, and Paul Holley. «Utilizing light unmanned aerial vehicles for the inspection of curtain walls: a case study». In: *Construction Research Congress 2016*. 2016, pp. 2651–2659 (cit. on p. 5).

[3] Stefan Winkvist, Emma Rushforth, and Ken Young. «Towards an autonomous indoor aerial inspection vehicle». In: *Industrial Robot: An International Journal* 40.3 (2013), pp. 196–207 (cit. on p. 5).

[4] Bruno Silveira, Roseneia Melo, and Dayana Bastos Costa. «Using uas for roofs structure inspections at post-occupational residential buildings». In: *International Conference on Computing in Civil and Building Engineering*. Springer. 2020, pp. 1055–1068 (cit. on p. 6).

[5] G Morgenthal and N Hallermann. «Quality assessment of unmanned aerial vehicle (UAV) based visual inspection of structures». In: *Advances in Structural Engineering* 17.3 (2014), pp. 289–302 (cit. on p. 6).

[6] Xiangyu Zhuo, Friedrich Fraundorfer, Franz Kurz, and Peter Reinartz. «Optimization of OpenStreetMap Building Footprints Based on Semantic Information of Oblique UAV Images». In: *Remote. Sens.* 10 (2018), p. 624. URL: https://api.semanticscholar.org/CorpusID:19235777 (cit. on p. 6).

[7] Michael Bown and Kevin Miller. «The Use of Unmanned Aerial Vehicles for Sloped Roof Inspections – Considerations and Constraints». In: *Journal of Facility Management Education and Research* 2.1 (Jan. 2018), pp. 12–18. ISSN: 2474-6630. DOI: 10.22361/jfmer/93832. eprint: https://meridian.allenpress.com/jfmer/article-pdf/2/1/12/2321206/jfmer_93832.pdf. URL: https://doi.org/10.22361/jfmer/93832 (cit. on p. 6).

[8] A. Spasov, D. Petrova-Antonova, and E. Hristov. «A COMPARISON STUDY ON DEEP LEARNING MODELS FOR BUILDING ROOFTOP CLASSIFICATION». In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-1/W2-2023 (2023), pp. 47–53. DOI: 10.5194/isprs-archives-XLVIII-1-W2-2023-47-2023. URL: https://isprs-archives.copernicus.org/articles/XLVIII-1-W2-2023/47/2023/ (cit. on p. 6).

[9] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. «Deep learning for visual understanding: A review». In: *Neurocomputing* 187 (2016). Recent Developments on Deep Big Vision, pp. 27–48. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2015.09.116. URL: https://www.sciencedirect.com/science/article/pii/S0925231215017634 (cit. on p. 6).

[10] Junyi Chai and Anming Li. «Deep Learning in Natural Language Processing: A State-of-the-Art Survey». In: *2019 International Conference on Machine Learning and Cybernetics (ICMLC)*. 2019, pp. 1–6. DOI: 10.1109/ICMLC48188.2019.8949185 (cit. on p. 6).

[11] Jian Huang, Junyi Chai, and Stella Cho. «Deep learning in finance and banking: A literature review and classification». In: *Frontiers of Business Research in China* 14.1 (2020), p. 13 (cit. on p. 6).

[12] Wahidya Nurkarim and Arie Wahyu Wijayanto. «Building footprint extraction and counting on very high-resolution satellite imagery using object detection deep learning framework». In: *Earth Science Informatics* 16.1 (2023), pp. 515–532 (cit. on p. 6).

[13] Mark Amo-Boateng, Nana Sey, Amprofi Amproche, and Martin Domfeh. «Instance segmentation scheme for roofs in rural areas based on Mask R-CNN». In: *Egyptian Journal of Remote Sensing and Space Science* (Apr. 2022). DOI: 10.1016/j.ejrs.2022.03.017 (cit. on p. 7).

[14] Jonguk Kim, Hyansu Bae, Hyunwoo Kang, and Suk Gyu Lee. «CNN Algorithm for Roof Detection and Material Classification in Satellite Images». In: *Electronics* 10.13 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10131592. URL: https://www.mdpi.com/2079-9292/10/13/1592 (cit. on p. 7).

[15] Youngseok Lee and Jongweon Kim. «Robustness of deep learning models for vision tasks». In: *Applied Sciences* 13.7 (2023), p. 4422 (cit. on p. 7).

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «Imagenet classification with deep convolutional neural networks». In: *Advances in neural information processing systems* 25 (2012) (cit. on p. 7).

[17] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. «Decaf: A deep convolutional activation feature for generic visual recognition». In: *International conference on machine learning*. PMLR. 2014, pp. 647–655 (cit. on p. 7).

[18] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. «Deep neural networks for object detection». In: *Advances in neural information processing systems* 26 (2013) (cit. on p. 7).

[19] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. «Face recognition: A convolutional neural-network approach». In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113 (cit. on p. 7).

[20] Xingcheng Luo, Ruihan Shen, Jian Hu, Jianhua Deng, Linji Hu, and Qing Guan. «A deep convolution neural network model for vehicle recognition and face recognition». In: *Procedia Computer Science* 107 (2017), pp. 715–720 (cit. on p. 7).

[21] Tara N Sainath, Brian Kingsbury, Abdel-rahman Mohamed, George E Dahl, George Saon, Hagen Soltau, Tomas Beran, Aleksandr Y Aravkin, and Bhuvana Ramabhadran. «Improvements to deep convolutional neural networks for LVCSR». In: *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE. 2013, pp. 315–320 (cit. on p. 7).

[22] Junqi Jin, Kun Fu, and Changshui Zhang. «Traffic sign recognition with hinge loss trained convolutional neural networks». In: *IEEE transactions on intelligent transportation systems* 15.5 (2014), pp. 1991–2000 (cit. on p. 7).

[23] Harry Pratt, Frans Coenen, Deborah M Broadbent, Simon P Harding, and Yalin Zheng. «Convolutional neural networks for diabetic retinopathy». In: *Procedia computer science* 90 (2016), pp. 200–205 (cit. on p. 7).

[24] Tyler Clark, Alexander Wong, Masoom A Haider, and Farzad Khalvati. «Fully deep convolutional neural networks for segmentation of the prostate gland in diffusion-weighted MR images». In: *Image Analysis and Recognition: 14th International Conference, ICIAR 2017, Montreal, QC, Canada, July 5–7, 2017, Proceedings 14*. Springer. 2017, pp. 97–104 (cit. on p. 7).

[25] David H. Hubel and Torsten N. Wiesel. «Receptive fields and functional architecture of monkey striate cortex». In: *The Journal of Physiology* 195 (1968). URL: https://api.semanticscholar.org/CorpusID:7136759 (cit. on p. 7).

[26] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. «Neocognitron: A neural network model for a mechanism of visual pattern recognition». In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 826–834. DOI: 10.1109/TSMC.1983.6313076 (cit. on p. 7).

[27] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. «Handwritten digit recognition with a back-propagation network». In: *Proceedings of the 2nd International Conference on Neural Information Processing Systems*. NIPS'89. Cambridge, MA, USA: MIT Press, 1989, pp. 396–404 (cit. on p. 7).

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 7).

[29] Hecht-Nielsen. «Theory of the backpropagation neural network». In: *International 1989 Joint Conference on Neural Networks*. 1989, 593–605 vol.1. DOI: 10.1109/IJCNN.1989.118638 (cit. on p. 7).

[30] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. «Parallel distributed processing model with local space-invariant interconnections and its optical architecture». In: *Appl. Opt.* 29.32 (Nov. 1990), pp. 4790–4797. DOI: 10.1364/AO.29.004790. URL: https://opg.optica.org/ao/abstract.cfm?URI=ao-29-32-4790 (cit. on p. 7).

[31] Xiao-Xiao Niu and Ching Y. Suen. «A novel hybrid CNN–SVM classifier for recognizing handwritten digits». In: *Pattern Recognition* 45.4 (2012), pp. 1318–1325. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2011.09.021. URL: https://www.sciencedirect.com/science/article/pii/S0031320311004006 (cit. on p. 7).

[32] Olga Russakovsky et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV]. URL: https://arxiv.org/abs/1409.0575 (cit. on p. 7).

[33] Jiuxiang Gu et al. *Recent Advances in Convolutional Neural Networks*. 2017. arXiv: 1512.07108 [cs.CV]. URL: https://arxiv.org/abs/1512.07108 (cit. on pp. 8, 15).

[34] Ki Bum Lee, Sejune Cheon, and Chang Ouk Kim. «A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes». In: *IEEE Transactions on Semiconductor Manufacturing* 30.2 (2017), pp. 135–142 (cit. on p. 8).

[35] 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 (cit. on p. 8).

[36] Kyle Gao, Mengge Chen, Sarah Narges Fatholahi, Hongjie He, Hongzhang Xu, José Marcato Junior, Wesley Nunes Gonçalves, Michael A. Chapman, and Jonathan Li. «A region-based deep learning approach to instance segmentation of aerial orthoimagery for building rooftop extraction». In: *Geomatica* 75.3 (2021), pp. 148–164. URL: https://cdnsciencepub.com/doi/abs/10.1139/geomat-2021-0009 (cit. on p. 8).

[37] Yuwei Cai, Hongjie He, Ke Yang, Sarah Narges Fatholahi, Lingfei Ma, Linlin Xu, and Jonathan Li. «A comparative study of deep learning approaches to rooftop detection in aerial images». In: *Canadian Journal of Remote Sensing* 47.3 (2021), pp. 413–431 (cit. on p. 8).

[38] Roberto Castello, Simon Roquette, Martin Esguerra, Adrian Guerra, and Jean-Louis Scartezzini. «Deep learning in the built environment: automatic detection of rooftop solar panels using Convolutional Neural Networks». In: *Journal of Physics: Conference Series* 1 (2019). URL: https://dx.doi.org/10.1088/1742-6596/1343/1/012034 (cit. on p. 8).

[39] L. Hang and G. Y. Cai. «CNN BASED DETECTION OF BUILDING ROOFS FROM HIGH RESOLUTION SATELLITE IMAGES». In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-3/W10 (2020), pp. 187–192. URL: https://isprs-archives.copernicus.org/articles/XLII-3-W10/187/2020/ (cit. on p. 8).

[40] Zoe Mayer, James Kahn, Yu Hou, Markus Götz, Rebekka Volk, and Frank Schultmann. «Deep learning approaches to building rooftop thermal bridge detection from aerial images». In: *Automation in Construction* 146 (2023), p. 104690. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2022.104690. URL: https://www.sciencedirect.com/science/article/pii/S092658052200560X (cit. on p. 8).

[41] David Zahradník, Filip Roučka, and Linda Karlovská. «Flat roof classification and leaks detections by Deep Learning». In: *Stavební obzor - Civil Engineering Journal* 32.4 (2024). DOI: 10.14311/CEJ.2023.04.0042. URL: https://ojs.cvut.cz/ojs/index.php/cej/article/view/9653 (cit. on p. 9).

[42] Chia-Cheng Yeh, Yang-Lang Chang, Mohammad Alkhaleefah, Pai-Hui Hsu, Weiyong Eng, Voon-Chet Koo, Bormin Huang, and Lena Chang. «YOLOv3-Based Matching Approach for Roof Region Detection from Drone Images». In: *Remote Sensing* 13.1 (2021). ISSN: 2072-4292. DOI: 10.3390/rs13010127. URL: https://www.mdpi.com/2072-4292/13/1/127 (cit. on p. 9).

[43] Yuchu Qin, Yunchao Wu, Bin Li, Shuai Gao, Miao Liu, and Yulin Zhan. «Semantic Segmentation of Building Roof in Dense Urban Environment with Deep Convolutional Neural Network: A Case Study Using GF2 VHR

Imagery in China». In: *Sensors* 19.5 (2019). DOI: 10.3390/s19051164. URL: https://www.mdpi.com/1424-8220/19/5/1164 (cit. on p. 9).

[44] Dmitry A. Yudin, Vasily Adeshkin, Alexandr V. Dolzhenko, Alexandr Polyakov, and Andrey E. Naumov. «Roof Defect Segmentation on Aerial Images Using Neural Networks». In: *Advances in Neural Computation, Machine Learning, and Cognitive Research IV*. Ed. by Boris Kryzhanovsky, Witali Dunin-Barkowski, Vladimir Redko, and Yury Tiumentsev. Cham: Springer International Publishing, 2021, pp. 175–183. ISBN: 978-3-030-60577-3 (cit. on p. 9).

[45] kumar_satyam. «Computer Vision Tutorial». In: *geeksforgeeks* (2024). URL: https://www.geeksforgeeks.org/computer-vision/ (cit. on p. 9).

[46] Amazon Web Services. *What is Computer Vision?* Accessed: 2024-10-09. 2024. URL: https://aws.amazon.com/what-is/computer-vision/ (cit. on pp. 9, 10).

[47] xis.ai. «A Complete Guide to Computer Vision: Types, Techniques, Applications». In: *Medium* (2024) (cit. on pp. 10, 12).

[48] Carolyn Joy V. «Understanding Semantic Segmentation Vs. Instance Segmentation for Object Recognition». In: *Linked AI* (2019). URL: https://www.linkedai.co/blog/understanding-semantic-segmentation-vs-instance-segmentation-for-object-recognition (cit. on pp. 10, 11).

[49] Adrian Rosebrock. *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017 (cit. on pp. 13, 14).

[50] Srivignesh Rajan. «An Introduction to Artificial Neural Networks». In: *Medium* (2020) (cit. on p. 14).

[51] Mark Scapicchio Jim Holdsworth. «What is deep learning?» In: *IBM* (2024). URL: https://www.ibm.com/topics/deep-learning (cit. on p. 14).

[52] saumyasaxena2730. «Introduction to Deep Learning». In: *GeeksforGeeks* (2024). URL: https://www.geeksforgeeks.org/introduction-deep-learning/ (cit. on pp. 14, 15).

[53] Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE]. URL: https://arxiv.org/abs/1511.08458 (cit. on pp. 15, 16).

[54] Zoumana KEITA. «An Introduction to Convolutional Neural Networks (CNNs)». In: *DataCamp* (2023). URL: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns (cit. on p. 15).

[55] goelaparna1520. «Convolutional Neural Network (CNN) in Machine Learning». In: *GeeksforGeeks* (2024). URL: https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/ (cit. on p. 15).

[56] University of Massachusetts Amherst. *Convolutional Networks*. URL: https://compsci682.github.io/notes/convolutional-networks (cit. on pp. 15–17, 59).

[57] Computer Vision Tutorial. «Convolutional Neural Network — Lesson 5: Strides». In: *Medium* (2023). URL: https://medium.com/@nerdjock/convolutional-neural-network-lesson-5-strides-2ffdeacf8f2c (cit. on p. 16).

[58] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556 (cit. on p. 18).

[59] François Chollet. *Xception: Deep Learning with Depthwise Separable Convolutions*. 2017. arXiv: 1610.02357 [cs.CV]. URL: https://arxiv.org/abs/1610.02357 (cit. on p. 18).

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385 (cit. on p. 18).

[61] Nitish Kundu. «Exploring ResNet50: An In-Depth Look at the Model Architecture and Code Implementation». In: *Medium* (2023). URL: https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f (cit. on p. 18).

[62] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV]. URL: https://arxiv.org/abs/1801.04381 (cit. on p. 19).

[63] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV]. URL: https://arxiv.org/abs/1911.11929 (cit. on p. 19).

[64] Gaurav Nair. «The Spark Your Neural Network Needs: Understanding the Significance of Activation Functions». In: *Medium* (2023) (cit. on pp. 19, 20).

[65] Sachinsoni. «Transfer Learning in Deep Learning from scratch». In: *Medium* (2023) (cit. on pp. 20, 21).

[66] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. *Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation*. 2018. arXiv: 1802.02611 [cs.CV]. URL: https://arxiv.org/abs/1802.02611 (cit. on p. 21).

[67] O. Ronneberger, P.Fischer, and T. Brox. «U-Net: Convolutional Networks for Biomedical Image Segmentation». In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. (available on arXiv:1505.04597 [cs.CV]). Springer, 2015, pp. 234–241. URL: http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a (cit. on p. 21).

[68] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV]. URL: https://arxiv.org/abs/1703.06870 (cit. on pp. 22, 26).

[69] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV]. URL: https://arxiv.org/abs/2408.15857 (cit. on p. 23).

[70] Range King. *Brief summary of YOLOv8 model structure _ Issue #189*. 2023. URL: https://github.com/ultralytics/ultralytics/issues/189 (cit. on p. 24).

[71] Kizito Nyuytiymbiy. «Parameters and Hyperparameters in Machine Learning and Deep Learning». In: *Towards Data Science* (2020) (cit. on p. 23).

[72] XQ. «Explained: Hyperparameters in Deep Learning». In: *Medium* (2024). URL: https://medium.com/the-research-nest/explained-hyperparameters-in-deep-learning-9b1e0f3b9029 (cit. on pp. 23, 24).

[73] Juan Terven, Diana M. Cordova-Esparza, Alfonso Ramirez-Pedraza, Edgar A. Chavez-Urbiola, and Julio A. Romero-Gonzalez. *Loss Functions and Metrics in Deep Learning*. 2024. arXiv: 2307.02694 [cs.LG]. URL: https://arxiv.org/abs/2307.02694 (cit. on pp. 25–27, 61).

[74] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497 (cit. on pp. 25, 26).

[75] Muhammad Yaseen. *What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector*. 2024. arXiv: 2408.15857 [cs.CV] (cit. on p. 26).

[76] Vineeth S Subramanyam. «IOU (Intersection over Union)». In: *Medium* (2021). URL: https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef (cit. on p. 26).

[77] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019 (cit. on p. 30).

[78] bubbliiiing. *deeplabv3-plus-pytorch*. URL: https://github.com/bubbliiiing/deeplabv3-plus-pytorch (cit. on p. 30).

[79] bubbliiiing. *unet-pytorch*. URL: https://github.com/bubbliiiing/unet-pytorch (cit. on p. 30).

[80] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *Ultralytics YOLOv8*. Version 8.0.0. 2023. URL: https://github.com/ultralytics/ultralytics (cit. on p. 30).

[81] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV] (cit. on p. 30).

[82] Junyi Chai, Hao Zeng, Anming Li, and Eric W.T. Ngai. «Deep learning in computer vision: A critical review of emerging techniques and application scenarios». In: *Machine Learning with Applications* 6 (2021), p. 100134. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000670 (cit. on pp. 59, 60).

[83] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. *Ultralytics YOLO*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics (cit. on p. 62).