Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research
University Centre of Illizi
Institute of Science

Department of Computer Science

Artificial Intelligence & its Applications Master's Thesis

Animal Detection in the Airport Area Via Artificial Intelligence: An Approach for Wildlife Hazard Prevention

Authors:

- Mr. BENFEDILA Houssem
  benfedilahsm@gmail.com

Advisors:

- Mr. MESSIKH Chouaib
  messikh.chouaib@cuillizi.dz

Academic Year: 2025

Abstract

Air transport, which is constantly being expanded, requires optimal airport security. Among the underdocumented but continuous risks is wildlife risk, which incurs collisions and high-order threats. Traditional solutions to dealing with this risk reflect limitations regarding accuracy and response time. Faced with such observation, this thesis explores the viability of deep learning in maximizing animal detection within airport environments, through proposing an accurate, swift, and adaptive solution.

Key words: wildlife hazard, artificial intelligence , Machine learning, Deep Learning, Artificial Neural Networks, Convolutional Neural Networks,Region-based Convolutional Neural Network, R-CNN, Fast Region-based Convolutional, Faster Region-Convolutional Neural Network, Yolov11, collision, Airport safety, Federal Aviation Administration, International Civil Aviation Organization.

Acknowledgements

I would like to express my sincere thanks and deep appreciation to my supervisor, Mr. MESSIKH Chouaib, for his continuous support, precise academic guidance, and constructive feedback throughout the completion of this work.

With deepest thanks and gratitude to my wife, Dr. Rim GASMI, whose unwavering support, constant encouragement, and valuable intellectual guidance formed a fundamental pillar throughout this academic project. Her patience and unwavering belief in me had a profound impact on the completion of this work, and I remain immensely grateful for her continuous support by my side.

I also extend my heartfelt gratitude to all the faculty members of the Master's program in Artificial Intelligence and its Applications at the University Center of Illizi, for the advanced knowledge and scientific tools they provided, which greatly helped me establish the necessary foundations for this research project.

I would like to express my great appreciation to the aeronautical safety experts and airport personnel who contributed their valuable insights and expertise regarding the risks posed by wildlife in airport environments. Their input played a key role in bridging the gap between theoretical research and practical application.

My deepest thanks and love go to my dear family, who have always been my greatest support and strength. Without their constant encouragement, patience, and prayers, this work would not have been possible.

I am also thankful to my friends and fellow students for their spirit of cooperation, mutual support, and encouragement, which added a human and collaborative dimension to this academic journey.

This research would not have come to light without the contributions and support of all these individuals and institutions. I am truly grateful to each and every one of them.

Table of Contents

List of Figures

List of Tables

Chapter 1

General Introduction

1.1 General Context

Air transport, the backbone of global mobility, has experienced consistent growth year after year in terms of passenger and cargo traffic. This activity imposes conditions on airport authorities for prevention from all accidents within the airport zone to ensure continuity and smoothness of air activities. At the heart of such concerns, an underestimated, but recurring, problem piques the interest of airport authorities due to its significance: wildlife hazard.

The risk of collision between aircraft and wildlife, particularly birds, can compromise air safety.

Wildlife hazard includes all risk concerning the presence of animals—mammals, birds, or others—in airport environments, especially on runways and taxiways, but also on circulatory paths of vehicles. These intrusions can lead to bird-aircraft crashes, "wildlife strikes", and potentially pose danger to ground personnel, especially when intrusions involve stray dogs, wolves, or foxes. Data compiled by the International Civil Aviation Organization (ICAO) and the Federal Aviation Administration (FAA) suggest that thousands of such events happen every year around the globe, leading to loss of material, delays in flights, and in some cases fatal accidents [1]

The causes are numerous: Proximity to their natural habitat (forests, water bodies), the appeal of organic waste for logistics centers, physical barrier degradation due to natural factors or animal and human interference, or environmental disturbances compelling animals to move. In response to this threat, the majority of airports have implemented safeguarding mechanisms such as sound devices, fencing, man patrols, and radar systems [2].

However, these traditional approaches suffer from several drawbacks: lack of precision, extremely slow response times, lack of continuous monitoring, excessive maintenance needs, and ineffectiveness in responding to specific adaptive animal behaviors. In these situations, the use of artificial intelligence (AI) technologies, and deep learning specifically, is turning out to be a viable alternative that can offer continuous, automatic, and precise monitoring.

Thanks to recent advances in convolutional neural networks (CNNs), advanced object detection systems are now able to identify, locate, and track animals in pictures or video streams in real-time, even in complex or dark environments[3]. Advanced systems such as these not only have the potential to improve the readiness of the airport security units but also to contribute to the deployment of more efficient preventive measures, significantly reducing the risks involved in wildlife threats.

It is within this kind of vision that the present research falls. we intend to design an artificial intelligence system for detecting animals in the airport area, utilizing the study and implementation of two modern object detection techniques: **Faster R-CNN** and **Yolov11n**.

1.2 Problem Statement

In a large and dynamic environment as economically and socially important as an airport, and especially on the airside area, the presence of animals poses various risks such as: collision with the vehicles or the aircraft, attacks on persons, and flight delays. Animal detection presents several challenges: the variability of species, the animal's camouflage in nature, the weather, and the animal's roaming time in the area. Traditional methods of observation, such as manual roving or conventional surveillance cameras and radars, prove to be inefficient at times and lead to latency in reaction at other times.

To what extent can deep learning improve the detection of animals within airport areas, leading to a decision that is precise, efficient, and adjustable enough to compensate for the specific constraints

of airports?

This thesis aims to answer these questions, evaluating the potential of deep learning to address current and future wildlife risk management challenges in civil aviation.

1.3 Objectives

The general objective of this thesis is to create and design a deep learning-based system for the detection of animals on runways and sensitive areas of airports, under any weather condition, and to provide data in order to enable the rapid intervention of specialists, with the objective of reducing the risks associated with wildlife threats.

The built system must satisfy the technical detection needs and the operational requirements of airport environments.

1.4 Thesis Outline

This thesis consists of four main chapters organized as follows:

- **Chapter 2: Animals hazard**
  We will examine the context of wildlife hazards in airports, defining the concept and its effects on air safety and ground personnel. We will identify the most concerned species (birds, deer, rodents, wolves, foxes, pigs, dogs), analyze incident statistics, their economic and human impacts, and the recent trends. The chapter also presents the traditional management methods (e.g. habitat modification, repellents, and radars), as well as the regulatory framework. Finally, it highlights the limitations of the classical approaches and the need for advanced solutions based on artificial intelligence, such as real-time detection models.

- **Chapter 3: Artificial Intelligence and Animal Detection**
  This chapter provides a comprehensive review of the state of the art in artificial intelligence techniques applied to object detection, in this case, wildlife detection in the airport environments. It explains the fundamentals of Convolutional Neural Networks (CNNs) and their main architecture. It also reviews real-time object detection algorithms like YOLO and Faster R-CNN, their performance, and their suitability for airport security. Finally, it presents the existing research and systems that uses AI for animan detection in the aviation, comparing the different approaches and their merits and demerits.

- **Chapter 4: Methodology**
  This chapter presents the methodology adopted to develop an intelligent system for detecting animals in an airport environment. It describes the system architecture, the data collection and annotation process (real, public, and synthetic images), as well as the augmentation techniques used to simulate various environmental conditions. A rigorous data splitting strategy is applied with particular attention to the diversity of species. Finally, the choice of YOLOv11n and Faster R-CNN models is justified based on their respective performance in speed and accuracy, in order to meet the requirements of real-time detection in complex environments.

- **Chapter 5: Implementation**
  This chapter documents the hands-on deployment and the testing of the two opted deep learning

models (Faster R-CNN and YOLOv11n) in detecting animals in the airport's airspace. It describes the computing environment on the Kaggle platform and the PyTorch and Ultralytics toolchains. Moreover, this chapter details the hyperparameter tuning process for the two models, highlighting the chosen configurations for optimal performance. It also outlines the key evaluation metrics used to quantify the accuracy, speed, and robustness of the models. The results section provides a general overview of the performance of the models on the test dataset, supported by training and validation loss and mAP curves. A comparative analysis is presented in order to highlight the strengths and weaknesses of Faster R-CNN and YOLOv11n, in addition to their practical implications for airport security. Finally, the chapter concludes each model's limitations based on the experimental findings and proposes potential optimizations and future research work directions, including model optimization, data augmentation, hybrid methods, and real-world testing.

Chapter 2

Animal hazard

2.1 Introduction

This chapter presents the problem of wildlife hazards at airports, their definition, effects, targeted animals (birds, deer, rodents, wolves, foxes, pigs, dogs), existing control measures, such as worldwide and national legislations, and the necessity of novel solutions through artificial intelligence (AI). Wildlife hazards, especially those associated with airplane-wildlife collisions, represent a great challenge to aviation safety, are very expensive, and expose ground personnel to danger. This chapter employs trustworthy sources and recent information.

2.2 Definition

Animal hazard or Wildlife strike refers to collisions between aircraft (airplanes, helicopters) and free-ranging animals, birds primarily but also mammals. They most frequently occur at low altitudes of flight, i.e., during takeoff, landing, or ground flight stages. According to the Federal Aviation Administration, birds are involved in approximately 97% of strikes, but mammals such as deer or pigs can cause disastrous damage through their weight and impact on the ground [1].

2.3 Context

Airports, which are most often found in the vicinity of natural habitats like wetlands, forests, or grasslands, are appealing to wildlife because of the presence of resources like food (food residues, seeds, insects), water (lakes, drainage ditches), and habitat (open spaces for migratory birds)[2]. Specifically, the runways, with their expansive cleared terrain, are a replica of the natural habitats of some animals like birds or rodents, thus the likelihood of encountering wildlife.

Moreover, the wildlife risks directly endanger the ground staff such as: the ramp agents, the baggage handlers, and the maintenance personnel. Stray dogs, attracted by garbage or abandoned near airports, can become vicious and bite the staff during ground operations, for example, while baggage is being loaded or during runway inspection. Wolves, present at a few airports close to wilderness regions, pose a threat by territorial or defensive actions, endangering crews carrying out activities such as repair or inspection of facilities. Foxes, which prey on rodents on runways, pose hazards by darting across taxiways unexpectedly and making personnel vehicles make sharp evasive turns, thus enhancing the possibility of accidents[4]. These encounters with animals subject personnel to physical harm (e.g., bites, vehicle collisions) and operational interference, requiring prompt action in high-risk areas (e.g., the zones in close proximity to active runways).

The growth in international air traffic, to some 38 million flights in 2023, as per the International Air Transport Association (IATA), and growing urbanization near airports raise hazards of wildlife collision and dangerous interaction for staff[5]. Climate change also modifies bird migration patterns and mammal behavior, creating additional threats in airport environs[2].

2.4 Statistics and Impacts

2.4.1 Frequency

Wildlife strikes are a global issue. The United States **National Wildlife Strike Database** records over 13,000 strikes per year for the period 1990 to 2023, totaling over 250,000. This is likely an underestimate, since minor strikes are often not reported. Approximately 47% of the strikes occur on landing, 31% during takeoff, and the remainder in flight or on the ground [1].

2.4.2 Economic Cost

Collisions are very expensive for the aviation industry. Direct damage (repair of airframe and engines) and indirect damage (delays, cancellations, inspections) account for between 650 million and 1 billion dollars per year in the United States alone, according to the Animal and Plant Health Inspection Service (APHIS). Worldwide costs can range into several billion dollars. For instance, a deer strike can result in landing gear damage with repair costs over $500,000, while bird ingestion into an engine can total over $1 million[6].

2.4.3 Safety

Wildlife hazards pose a threat to both ground personnel and in-flight safety. Over 200 deaths have been recorded globally since 1988 due to in-flight collisions, the FAA states, at approximately 1 fatal accident per billion flight hours. The US Airways Flight 1549 incident (2009), in which a collision with Canada geese made it necessary to make an emergency landing on the Hudson River, is one example of the potential threat to crew and passengers[1].

In addition, wildlife represents a considerable hazard to ground personnel such as maintenance technicians, baggage handlers, and air traffic controllers. Collisions with mammals such as deer, pigs, or feral dogs on runways or taxiways may result in:

- **Physical injuries:** Large animal (pig or deer, for example) collisions while driving or conducting manual inspections can result in serious injuries like fractures or concussions.

- **Operational disruptions:** Unforeseen incursions by wildlife necessitate prompt action, exposing workers to harm (e.g., working in close proximity to active runways).

- **Psychological stress :** Repetitive exposures to potentially threatening wildlife, for example, wolves or stray canines, might increase stress and reduce the workers' productivity levels. A 2019 incident at a U.S. airport saw a maintenance worker injured when a vehicle swerved to avoid a deer, highlighting the need for improved wildlife detection systems. These hazards form the basis for the need for active wildlife management to protect aircraft as well as ground workers.

Aspect	Detail
Frequency	More than 13,000 collisions per year (United States, 1990–2023)
Economic Cost	$650 million to $1 billion per year (United States)
Safety	Over 200 deaths since 1988, approximately 1 fatal accident per billion flight hours; injuries to ground personnel

Table 2.1: Summary of Collision Statistics and Impacts in the United States

2.4.4 Species Involved

Seven animal species — deer, birds, rodents, wolves, foxes, pigs, and dogs — have been selected for the frequency or severity of their impact on airports.

2.4.4.1 Birds

Birds, representing 97% of the documented collisions, include species like:

- Canada Geese (Branta canadensis): Large migratory birds causing heavy damage.
- Pelicans (Pelecanus spp.): Occur along coasts.
- Cranes (Grus spp.): Large migratory birds.
- Gulls (Larus spp.): Attracted to waste at airports.

2.4.4.2 Mammalia

- Deer (Odocoileus spp.): Invade runways, causing severe ground impact and risk to ground personnel.
- Rodents (Rattus spp., Sciurus spp.): They are attracted by waste and may destroy electrical equipment.
- Wolves (Canis lupus): Found in airports bordering wilderness regions, potentially aggressive to workers
- Foxes (Vulpes vulpes): Prey on rodents along runways with minimal risk to staff.
- Sus scrofa pigs (wild boars) intrude into weakly secured places, with serious effects and hazard to ground staff.
- Dogs (Canis familiaris, feral): Drawn to rubbish or abandoned sites, constituting hazards to personnel engaged in ground operations.

2.4.5 Trends

Populations of many species are increasing. Among the 14 largest bird species in North America, 13 have expanded in the past 30 years due to conservation initiatives (e.g., wetlands preservation), APHIS reports. Deer and feral pig populations are also expanding in some regions, and feral dogs are a growing concern at developing-world airports[6].

2.5 Current Measures

2.5.1 Participating Organizations

- **Animal and Plant Health Inspection Service (APHIS):** Provides services through Wildlife Services, using biologists to assess and mitigate wildlife threats [6].
- **Federal Aviation Administration (FAA):** Sets safety standards and funds research[1].
- **International Civil Aviation Organization (ICAO):** Global organization producing international airport wildlife management standards (Annex 14)[7].
- **Wildlife Services:**Partners with airports for effective solutions[6].

2.5.2 Management Strategies

- **Habitat Modification:**Removal of food sources (waste disposal) and attractive habitats (draining wetlands).
- **Repellents:**Utilizing sonic (gas cannons), visual (lasers), or chemical devices to move wildlife away.
- **Exclusion Fencing:**Fencing to exclude mammals like deer or pigs.
- **Radar Surveillance:**Technology like Avian Radar to detect birds, although limited by finances and weather[2].

2.5.3 Regulations

Wildlife hazard management is governed nationally and globally:

- **ICAO Annex 14, Volume I (Aerodromes):** Affects airports to have wildlife management programs to evaluate and manage hazards, such as monitoring habitats on a repeated basis and species control programs[7].
- **FAA 14 CFR Part 139.337 (Wildlife Hazard Management):** Requires certified U.S. airports to possess and conduct Wildlife Hazard Assessments and design Wildlife Hazard Management Plans, including proactive and response measures to protect aircraft and ground personnel[8].
- **Migratory Bird Treaty Act (1918):**Protects migratory birds, restricting lethal control methods (e.g., culling) except with special authorization[9].
- **ICAO Airport Services Manual, Part 3 (Wildlife Control and Reduction):**Provides standards for habitat maintenance, detection equipment, and staff training to ensure safety for both aircraft operations and ground staff[10].
- **National Environmental Policy Act (NEPA):** Requires environmental assessments for wildlife control measures that may impact local ecosystems[2].

2.6 Necessity for Advanced Solutions

2.6.1 Disadvantages of Traditional Methods

- Thrice the Cost: Radar systems and secure fencing cost millions to install and maintain.
- Low Flexibility: Animals become accustomed to repellents (e.g., sonic cannons), reducing effectiveness.
- Poor Coverage: Manual or radar systems do not scan expansive airport land round-the-clock.
- Growing Threats: With a projected 44 million flights by 2030 (IATA) and a rising animal population, collisions and attacks on ground staff are growing [5].

2.6.2 Role of Artificial Intelligence

AI, as embodied by models like YOLOv11n and Faster R-CNN, offers thrilling options:

- Real-Time Detection: Continuous monitoring of runways and environments by cameras to protect planes and ground personnel.
- Better Accuracy: Targeting of small mammals or birds in adverse conditions (night, mist).
- Automation: Avoidance of operational cost and exposure of personnel to hazardous wildlife encounters.
- Integration: Can be integrated with airport control systems for immediate alert of ground personnel.

In conclusion, wildlife hazards pose a significant threat to aviation, threatening aircraft safety, economic stability, and ground staff welfare. AI has the potential to transform risk management, enhancing safety and reducing costs.

Chapter 3

Artificial Intelligence and Animal Detection

3.1 Introduction

Artificial Intelligence (AI) has been a game-changing technology that has radically changed multiple industries, including aviation, where AI is being utilized for solving serious issues like wildlife threats at airports. AI-driven solutions, more specifically computer vision-based solutions, present unmatched real-time detection and mitigation of animal threats. This chapter presents a detailed study of deep learning methods and their real-world applications. By exploring state-of-the-art models like **Faster R-CNN** and **YOLOv11n** for animal detection in the airport setting, this chapter sets the theoretical groundwork and practical context for exploring how AI can be used to improve aviation safety.

3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) can be seen as an evolution of traditional ANNs. They retain the fundamental concept of neurons that learn and adjust their parameters (weights) to optimize a score function (from the input image to the class prediction). The training techniques and loss functions used in classic ANNs are also applicable to CNNs.

The crucial difference is the adaptation of CNNs to image processing. Their architecture is designed to exploit the spatial structure of images, which allows them to learn relevant features more efficiently and with fewer parameters than traditional ANNs.

Traditional ANNs show their limitations in the face of the complexity of image data. The example of MNIST (28 x 28, 784 weights per initial neuron) contrasts sharply with a 64 x 64 color image (12,288 weights per initial neuron). This explosion in the number of parameters, combined with the need for larger networks, makes traditional ANNs impractical for complex vision tasks. CNNs, thanks to their specific design, largely overcome these limitations.

3.2.1 Architecture of CNN

In a classic neural network, each neuron in a layer is connected to all the neurons in the previous layer. The intermediate layers, located between the input and the output, are designated as hidden layers. These hidden layers are made up of several neurons, each linked to all the neurons in the layer that precedes it. However, the dense architecture [11] of fully connected networks poses a problem: it is poorly suited to processing large images. For this type of data, convolutional neural networks (CNNs) are generally preferred.

Designed to analyze gridded data, such as 1D time series, 2D or 3D images or voice signals, or 4D videos, the convolutional network is a deep neural network architecture. Its main characteristics include: a local receptive field, which limits connections to a small area of the input; weight sharing, which reduces the number of parameters; and subsampling (or pooling), which reduces spatial resolution while preserving essential information.

1. **Local Receptive Field** In a classic neural network, each neuron or hidden unit is connected to all the neurons in the previous layer or to all the input units. In contrast, convolutional neural networks (CNNs) adopt a local receptive field architecture, where each hidden unit interacts only with a small region of the input, called the local receptive field. This is achieved using a filter or weight matrix smaller than the input. Thanks to this approach, neurons can identify fundamental visual characteristics such as edges, corners, or end points.

2. **Weight Sharing** Weight sharing consists of applying the same filter or the same weights to all the receptive fields in a layer. In a convolutional network, the filters, smaller than the input, are used at each position of the image, which means that the same filter is shared for all local receptive fields. A convolutional network is made up of a succession of layers having distinct roles. A typical CNN architecture includes the following layers:

   (a) Convolutional layer: extracts features using filters.

   (b) Activation function layer (ReLU): introduces non-linearity.

   (c) Pooling layer: reduces the spatial size of the data.

   (d) Fully connected layer: produces class scores at the end of the network.

   (e) Dropout layer: prevents overfitting.

   These layers are stacked to form a complete CNN architecture. The convolutional and activation layers are generally grouped together, followed by an optional pooling layer. The last layer, fully connected, generates the classification scores of the input image. Furthermore, optional layers such as batch normalization can be added to speed up training, or the dropout layer to limit overfitting.

3. **Subsampling (Pooling)** Subsampling reduces the spatial size of the input, which decreases the number of parameters in the network. Among the subsampling techniques, the most common is max-pooling, which retains the most significant values in a given region while reducing the resolution.

3.2.2 CNN Layers

3.2.2.1 Convolutional layer:

The convolutional layer is the fundamental element of a convolutional neural network (CNN). It uses the convolution operation (denoted by *) instead of the classic matrix multiplication. Its parameters consist of a set of learnable filters, also called kernels. The main function of the convolutional layer is to identify characteristics present in local regions of the input image, common to the entire data set, and to map their presence in a feature map. Each filter in the layer generates a feature map by repeatedly applying the filter to sub-regions of the entire image, that is, by performing a convolution.

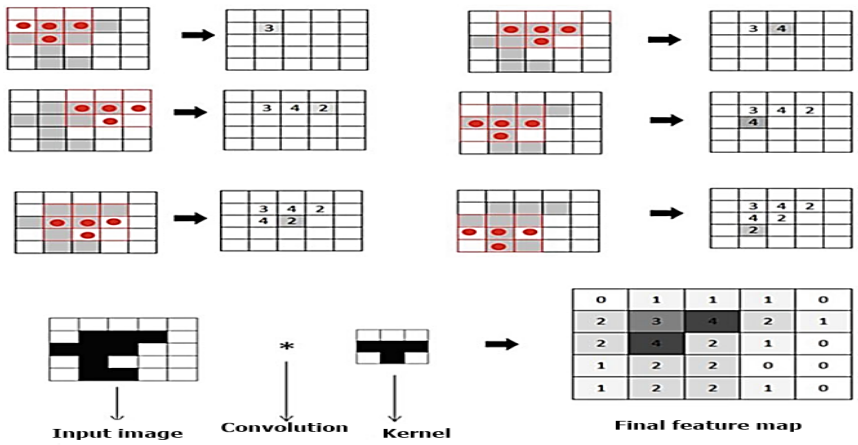

Figure 3.1: Exemple of convolution operation

The filter is applied to the input image, a bias term is added, and then an activation function is used. The area of the image on which the filter is applied is called the local receptive field, whose size corresponds to that of the filter. The feature map is obtained after adding a bias term and applying a non-linear function to the result of the convolution operation. The objective of this non-linear function is to introduce non-linearity into the convolutional network (ConvNet) model. Several non-linearity functions are available, and they will be briefly explained in the following section.

- **Filters/Kernels** The weights of each convolutional layer define the convolution filters, and a layer can contain several filters. Each filter captures a specific characteristic, such as an edge or a corner. During the forward pass, each filter is moved over the width and height of the input, thus generating a feature map specific to that filter. The hyperparameters of the convolutional layer of a convolutional neural network (ConvNet) influence both the quality of the results and the performance in terms of execution time and memory cost. The four essential hyperparameters of the convolutional layer are described below:

  1. Filter size: filters can have any size greater than 2×2 and less than that of the input, but common sizes generally vary from 11×11 to 3×3. The size of a filter is independent of that of the input.
  2. Number of filters: the number of filters can be chosen freely, within reasonable limits. For example, AlexNet uses 96 filters of size 11×11 in its first convolutional layer, while VGGNet uses 96 filters of size 7×7, and another variant of VGGNet employs 64 filters of size 11×11 in the first layer.
  3. Stride: this is the number of pixels the filter moves at each step to define the local receptive field. A stride of 1 means that the filter moves one pixel at a time, horizontally and vertically. Too small a stride results in significant overlap of the receptive fields, while too large a stride reduces overlap and decreases the spatial dimensions of the output volume.
  4. Zero Padding: this hyperparameter indicates the number of zero pixels to add around the input image. Zero padding makes it possible to control the spatial size of the output volume. Each filter in a convolutional layer generates a feature map whose size is calculated by the formula (([A - K + 2P]/S) + 1), where:

- (A) is the size of the input volume.
- (K) is the size of the filter.
- (P) is the number of padding pixels,
- (S) is the stride.

3.2.2.2 Activation Function

The output of each convolutional layer is passed to an activation function layer. This layer applies an activation function that takes the feature map generated by the convolutional layer and produces an activation map as output. The activation function transforms the activation level of a neuron into an output signal, thus defining the response of a neuron to a given input. It generally exerts a compression effect, taking an input (a number), performing a mathematical operation on it, and generating an activation level of the neuron within a specific range, for example between 0 and 1 or between -1 and 1.

A typical activation function must be differentiable and continuous over its entire domain. Since convolutional networks (ConvNets) are trained using gradient-based methods, the activation function must be differentiable at every point. However, if a non-gradient-based method is used, differentiability is not required. Numerous activation functions are used in artificial neural networks (ANN), and among the most common are the following:

- **Logistic/Sigmoid Activation Function:** The sigmoid function is mathematically defined by the following formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.1}$$

  It produces an S-shaped curve, as illustrated in Figure 3.2. The sigmoid function compresses input values into the interval [0, 1].

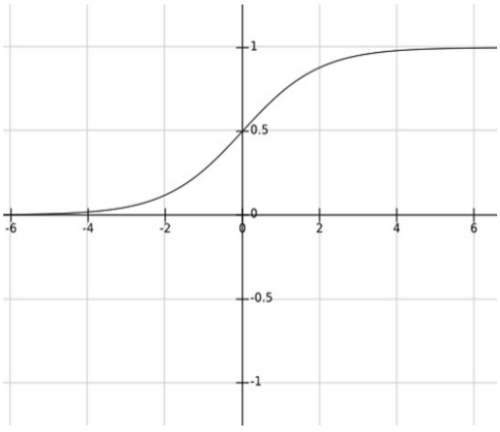

Figure 3.2: Graph of sigmoid activation function

- **Tanh Activation Function:** The hyperbolic tangent function (tanh) is similar to the sigmoid function, but its output lies in the interval [-1, 1]. The advantage of the tanh function over the sigmoid is that negative inputs are strongly mapped to negative values, and inputs close to zero are mapped close to zero on the graph of the tanh function, as illustrated in Figure 3.3.

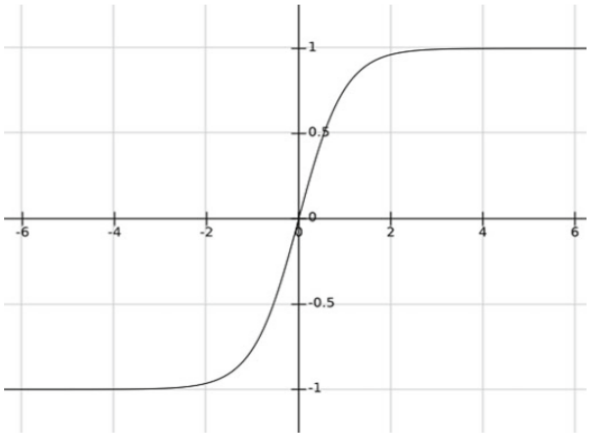

Figure 3.3: Graph of tanh activation function

- **Softmax Function (Exponential Function):** The softmax function is frequently used in the output layer of a neural network for classification tasks. It is mathematically defined by the following formula:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{pour} \quad i = 1, \ldots, K \tag{3.2}$$

  The softmax function is a generalization of the logistic activation function, adapted for multi-class classification.

- **ReLU Activation Function** The rectified linear unit (ReLU) has gained significant importance in recent years and is currently the most used activation function in deep neural networks. Networks using ReLU train much faster than those using other activation functions like sigmoid or tanh. The ReLU function calculates the activation by applying a threshold at zero: if the input is less than 0, the output is 0; otherwise, the output is equal to the input. Mathematically, it is expressed as follows:

$$f(x) = \max(0, x) \tag{3.3}$$

  The ReLU activation function produces a graph that is zero when x ¡ 0 and linear with a slope of 1 when x ¿ 0, as illustrated in Figure 3.4.

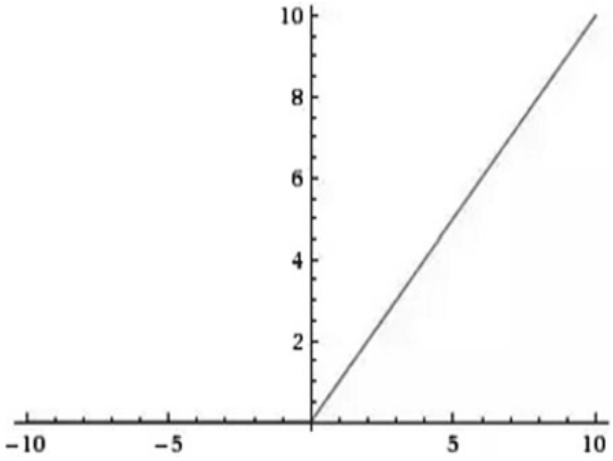

Figure 3.4: Rectified Linear Unit (ReLU) activation function

- **SWISH Activation Function** The SWISH activation function is a variant of the sigmoid function and is defined by the following formula:

$$f(x) = x \cdot \sigma(x) = x \cdot \frac{1}{1 + e^{-x}} \tag{3.4}$$

where σ(x) represents the sigmoid function. The SWISH activation function is non-monotonic, and its graph is shown in Figure 3.5.

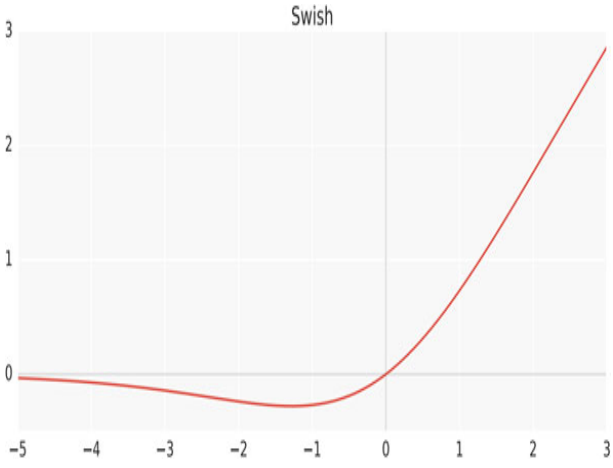

Figure 3.5: The Swish activation function

3.2.2.3 Pooling Layer

In convolutional neural networks (ConvNets), the sequence formed by a convolutional layer and an activation function layer is often followed by an optional pooling or subsampling layer, which reduces the spatial size of the input, thereby decreasing the number of parameters in the network. The pooling layer takes each feature map produced by the convolutional layer and subsamples it, that is, it summarizes a region of neurons from the convolutional layer. Several pooling techniques exist, the most common being max-pooling. Max-pooling simply selects the maximum value in the input region, which is generally a subpart of the input (for example, of size 2×2). For example, for an

input region of 2 × 2, the max-pooling unit returns the maximum value among the four values, as illustrated in Figure 3.6. Other pooling options include average pooling and L2-norm-based pooling.

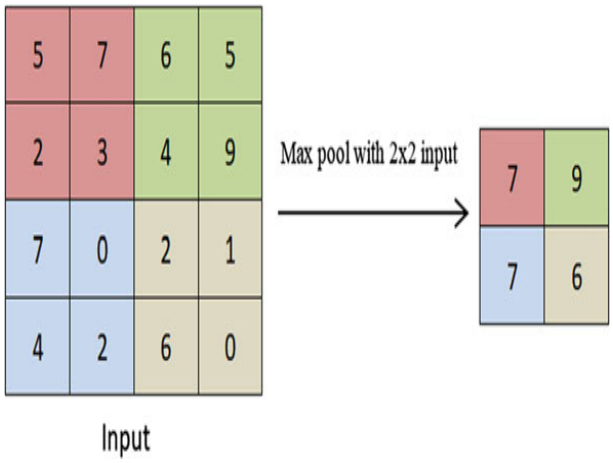

Figure 3.6: Max-pooling

3.2.2.4 Fully-connected layer

Convolutional neural networks (ConvNets) are structured in two main stages: feature extraction and classification. In ConvNets, the set of convolutional and pooling layers constitutes the feature extraction stage, while the classification stage consists of one or more fully connected layers followed by a softmax function layer. The convolution and pooling process continues until a sufficient number of features is detected. The next step is to make a decision based on these detected features. In the case of a classification problem, the features extracted in the spatial domain are used to calculate the probabilities that these features represent each class, that is, to obtain a class score. This is done by adding one or more fully connected layers at the end of the network. In a fully connected layer, each neuron in the previous layer (whether it is a convolutional, pooling, or fully connected layer) is connected to each neuron in the next layer, and each value contributes to predicting the extent to which it corresponds to a specific class. Figure 3.7 illustrates the connection between a convolutional layer and a fully connected layer. Like convolutional layers, fully connected layers can be stacked to learn even more complex combinations of features. The output of the last fully connected layer is passed to a classifier that produces the class scores. The two main classifiers used in ConvNets are the ”softmax” function and support vector machines (SVMs). The softmax classifier generates probabilities for each class, with a total probability sum equal to 1. In contrast, the SVM produces class scores, and the class with the highest score is considered the correct class.

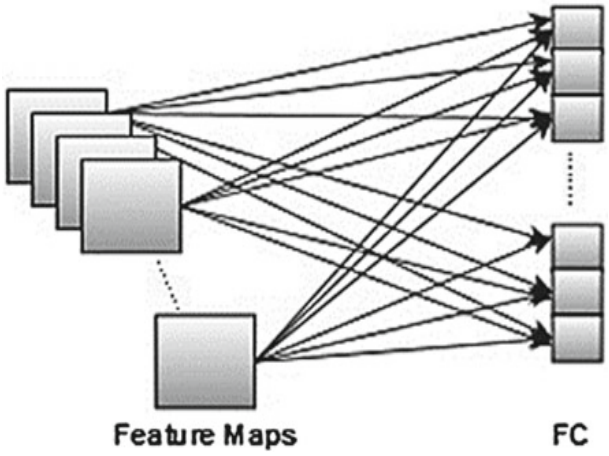

Figure 3.7: Connection between convolution layer and fully connected layer

3.2.2.5 Dropout

Deep neural networks have several hidden layers, which allow them to learn complex features. These layers are followed by fully connected layers for decision-making. A fully connected layer is connected to all the features, making it susceptible to overfitting. Overfitting occurs when a model is so well-trained on the training data that it performs poorly on new data. To remedy this problem, a dropout layer can be integrated into the model. This technique involves randomly removing some neurons and their connections during training (see Figure 3.8). This results in a reduced network, where the incoming and outgoing connections of the removed neurons are also eliminated. At this stage, only the reduced network is trained on the data. The removed neurons are then reintroduced into the network with their initial weights. Dropout significantly reduces overfitting and improves the model's ability to generalize.

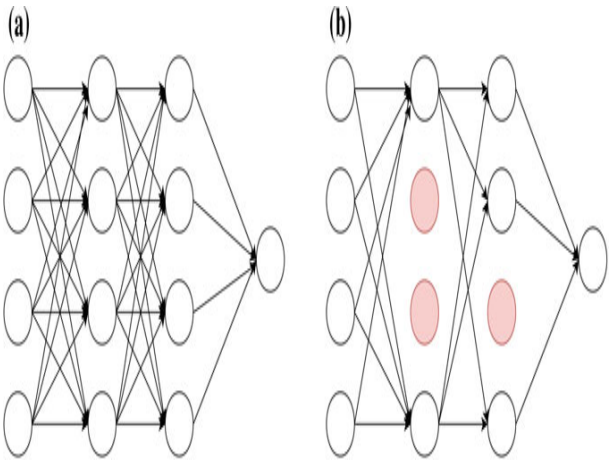

Figure 3.8: 'a' A simple neural network, 'b' neural network after dropout

3.2.3 Training CNNs

A well-trained artificial neural network has weights that amplify the relevant signal while attenuating noise. A larger weight indicates a stronger correlation between a signal and the result of the

network. The inputs associated with high weights influence the network's interpretation of the data more than those associated with lower weights. The learning process, for any weight-based learning algorithm, consists of readjusting the weights and biases, reducing some and increasing others. This makes it possible to assign importance to certain information while minimizing the impact of other information. This mechanism helps the model to identify which predictors (or features) are related to which results, and to adjust the weights and biases accordingly.

In most datasets, certain features are strongly correlated with certain labels (for example, the area in square meters is related to the selling price of a house). Neural networks discover these relationships blindly by making an estimate based on the inputs and the weights, and then evaluating the accuracy of the results. The loss functions in optimization algorithms, such as stochastic gradient descent (SGD), reward the network for good estimates and penalize it for bad ones. SGD adjusts the network parameters to favor correct predictions and move away from erroneous predictions.

3.2.3.1 Loss Functions and Softmax Classifier

A loss function [11] is used to calculate the error (the difference between the prediction and the actual label) during the training process of a deep neural network. Depending on the application, various loss functions can be used in deep learning networks. For example, the mean squared error (L2) loss, the cross-entropy loss, and the hinge loss are commonly used for classification problems. The absolute error (L1) loss is suitable for regression problems. Some of the most frequently used loss functions are presented below.

- **Mean Squared Error (L2) Loss** The most commonly used loss function in machine learning is the mean squared error (MSE) loss, also called the L2 loss function. This function calculates the mean squared error EEE of all the individual errors, and is expressed by the following formula:

$$E = \frac{1}{n} \sum_{i=1}^{n} e_i^2 \tag{3.5}$$

  Where e_i represents the individual error of the i-th output neuron, calculated as:

$$e_i = \text{target}(i) - \text{output}(I) \tag{3.6}$$

  During the training process, a loss function is applied to the output layer to calculate the error, and its derivative (gradient) is propagated backward through the network. The weights of the network are then updated according to their respective gradients.

- **Cross-Entropy Loss** The cross-entropy loss is another loss function frequently used in regression and classification problems. It is expressed by the following formula:

$$H(y) = -\sum_{i} y_i' \log(y_i) \tag{3.7}$$

  The Cross-entropy loss is another loss function frequently used in regression and classification problems. It is expressed by the following formula:

- **Softmax Classifier** The softmax classifier [11] is a mathematical function that takes a vector as input and produces an output vector whose values are between 0 and 1, and whose sum of

the elements is equal to 1. In other words, the sum of all the outputs of the softmax function is 1. The softmax function is defined by the following formula:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{3.8}$$

Since the softmax function generates a probability distribution, it is particularly useful in the final layer of deep neural networks for multiclass classification tasks.

3.2.3.2 Gradient Descent-Based Optimization Techniques

Gradient descent is an optimization technique used to minimize or maximize a cost function by calculating the gradients necessary to update the values of the network's parameters. Different variations of this optimization technique define how the parameter updates are calculated from these gradients.

- **Gradient Descent Variants** There are three commonly used variants of gradient descent (GD). These variants are distinguished by the number of training examples used to calculate the gradient. The three variants are:

- **Batch Gradient Descent (GD)** In traditional gradient descent, also called batch gradient descent, the gradient of the error with respect to the weight parameter w is calculated over the entire set of training data, and then the weights are updated according to the formula:

$$\mathbf{w} = \mathbf{w} - \mu \cdot \nabla E(\mathbf{w}) \tag{3.9}$$

Where ∇E(w) represents the gradient of the error with respect to w, and μ is the learning rate, a hyperparameter that determines the step size during the update. This learning rate must be neither too high (at the risk of missing the optimal minimum), nor too low (at the risk of greatly slowing down learning). However, when working with large datasets containing hundreds or thousands of examples, the calculation of the gradient becomes very costly in memory and time, which considerably slows down the training of the model.

- **Stochastic Gradient Descent (SGD)** This problem can be solved by using stochastic gradient descent (SGD), also called incremental gradient descent. Unlike classical gradient descent, SGD calculates the gradient of the error from a single training example at a time, followed immediately by an update of the parameters. The update is done according to the formula:

$$\mathbf{w} = \mathbf{w} - \mu \cdot \nabla E(\mathbf{w}; \mathbf{x}(i); \mathbf{y}(I)) \tag{3.10}$$

Where ∇E(w ; x(i), y(i)) represents the gradient of the loss function with respect to the parameters w, calculated for the training example x(i), y(i).
This method is generally faster because it performs an update for each example. However, this causes frequent fluctuations in the loss function, due to the instability caused by the individual updates.

- **Mini-batch gradient descent** Mini-batch gradient descent (or mini-batch SGD) is a hybrid method combining the advantages of classical gradient descent and stochastic descent. It

consists of dividing the training set into mini-batches of n examples, and then updating the parameters after each mini-batch, according to the formula:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \cdot \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \mathcal{L} \left(f(x_i; \theta^{(t)}), y_i \right) \tag{3.11}$$

This method is widely used in deep learning because it balances accuracy and efficiency. The typical size of a mini-batch generally varies between 50 and 256, and must be chosen carefully based on several factors:

- Large batch sizes: they give more precise gradients, but require a lot of memory.
- Small batch sizes: they have a regularizing effect, but introduce a high variance in the gradient calculation. This requires using a lower learning rate to stabilize training, which can slow down learning

3.3 Region-based Convolutional Neural Network (R-CNN)

To understand how R-CNN [12] works, it is essential to be familiar with the basic networks, whose performance depends on their architecture, such as LeNet, AlexNet, ZF Net, GoogLeNet, VGGNet, ResNet, and other similar architectures that are regularly evolving. The R-CNN architecture unfolds in three steps, as illustrated in Figure 3.9. First, an input image is analyzed to identify probable objects using an algorithm called Selective Search, which generates approximately 2,000 region proposals. Next, a convolutional neural network (CNN) is applied to each of these region proposals. Finally, the output of each CNN is used as input for a support vector machine (SVM) and a linear regressor. The SVM classifies the region corresponding to an object in the image, while the linear regressor precisely adjusts the bounding box of the detected object.
However, R-CNN has several major drawbacks:

1. High training time: classifying approximately 2,000 region proposals per image requires considerable time to train the network.
2. Impossible real-time detection: object detection takes about 47 seconds per test image, making the approach unsuitable for real-time applications.
3. Complex training: R-CNN requires separate training of three distinct models: the CNN to extract image features, the classifier to predict the class, and the regressor to refine the bounding boxes. This pipeline makes training particularly difficult.

Due to these limitations, improvements to R-CNN are necessary.

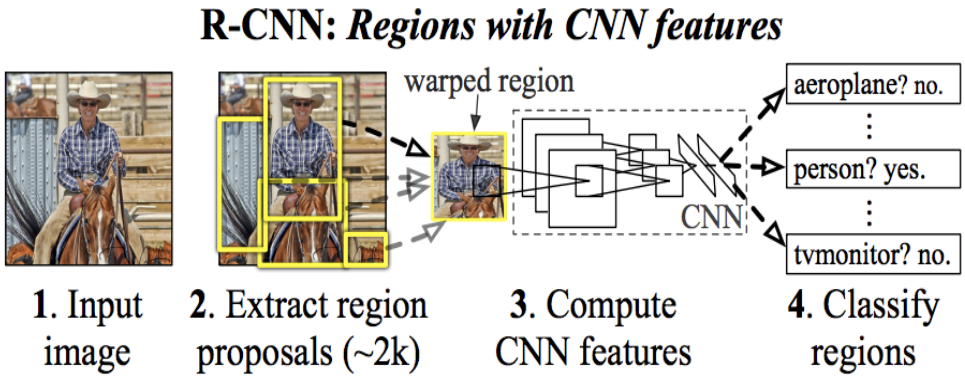

Figure 3.9: R-CNN architecture

3.4 Fast Region-based Convolutional Neural Network (Fast R-CNN)

The Fast R-CNN [12] approach is similar to that of R-CNN, but with a notable difference: instead of providing the region proposals directly to the CNN, the input image is passed to the CNN to generate a convolutional feature map. From this map, region proposals are identified and transformed into squares. A Region of Interest (RoI) pooling layer resizes them to a fixed size to pass them to a fully connected layer. To predict the class of the proposed regions, a softmax layer is used instead of an SVM, as illustrated in Figure 3.10:

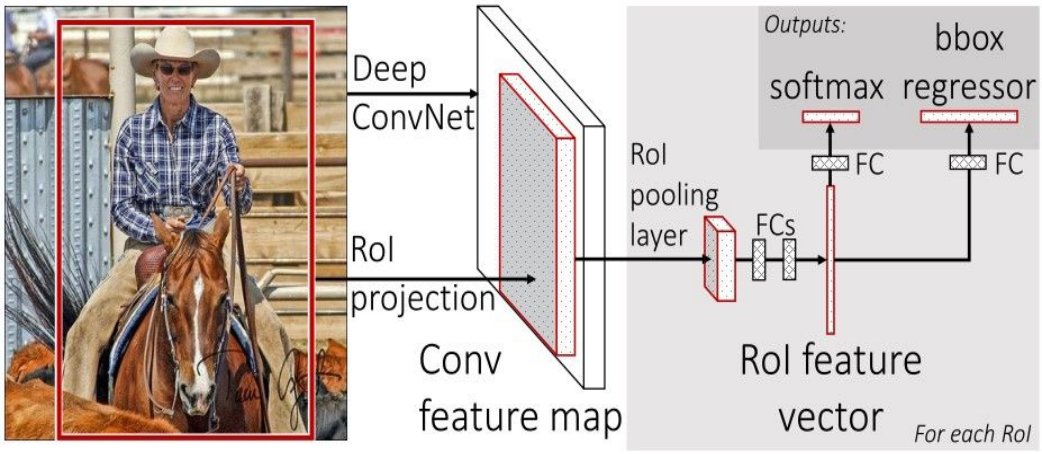

Figure 3.10: Fast R-CNN architecture

This algorithm is faster than R-CNN because it is not necessary to provide approximately 2,000 region proposals to the CNN each time. The image is processed only once to generate the feature map. In addition, the training of the CNN, the classifier, and the bounding box regressor in Fast R-CNN is performed simultaneously within a single model.
Despite these advantages, the main problem with this technique lies in the generation of region

proposals. These proposals, made using the Selective Search method, are slow and constitute a bottleneck in the overall process.

3.5 Faster Region-based Convolutional Neural Network (Faster R-CNN)

The slow selective search method, used for region proposals in Fast R-CNN, was replaced by a Region Proposal Network (RPN), giving rise to Faster R-CNN [12]. As illustrated in Figure 3.10, in Faster R-CNN, the input image is passed to the convolutional network (CNN). The RPN operates on the last layer of the CNN, where the feature map is projected into a lower dimension, for example, 256 dimensions, using a 3×3 sliding window called an anchor.

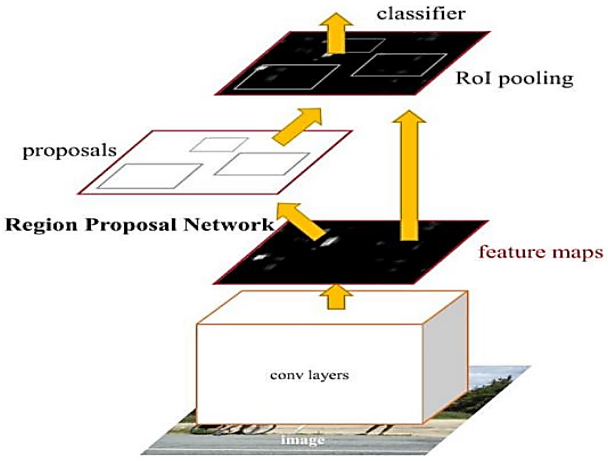

Figure 3.11: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network

The RPN [13] generates different types of anchor boxes based on the regions, with a fixed ratio (k) (Figure 3.11) for each position of the sliding window, such as a wide box, a tall box, or a square box. For each anchor box, if the intersection over union (IoU) is greater than 0.7, an object is detected; if it is less than 0.3, no object is present. An "objectness" score is thus calculated to propose a region containing an object.

Next, the region proposals are passed to a Region of Interest (RoI) pooling layer, followed by fully connected layers. Finally, the output is sent to a softmax classification layer and a bounding box regressor.

Faster R-CNN manages to accurately locate the position of objects in an image while offering good precision.

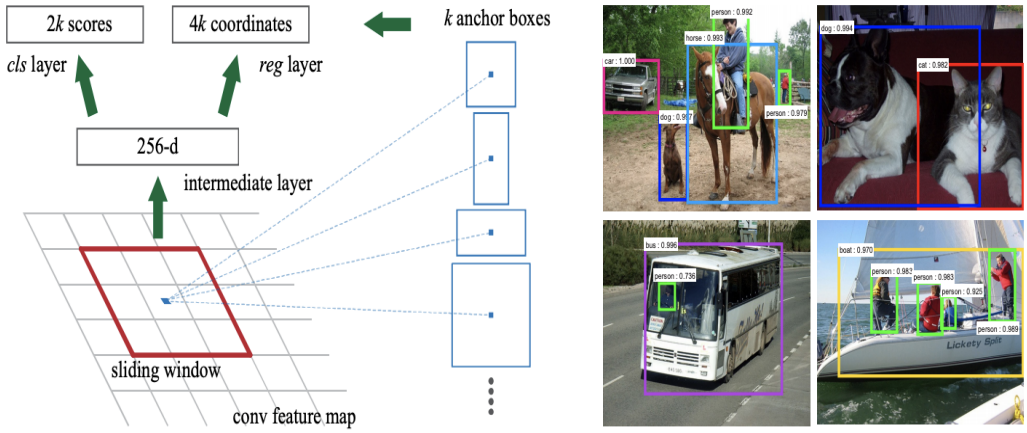

Figure 3.12: Left: Region Proposal Network (RPN). Right: Example detections using RPN proposals on PASCAL VOC 2007 test.

3.6 You Only Look Once (YOLO)

YOLO (You Only Look Once) [14]is a family of real-time object detection algorithms for video or images. Unlike two-stage approaches (such as R-CNN, which first propose regions and then classify them), YOLO classifies objects in a single pass, making it fast and suitable for real-time use in cases such as video monitoring, self-driving cars, or robotics [1]. YOLO defines object detection as a regression task, simultaneously predicting bounding boxes, confidence, and class probabilities.

In this approach, the authors reframe object detection as a regression problem instead of a classification task, spatially separating bounding boxes and assigning probabilities to each detected object using a single convolutional neural network (CNN).

3.6.1 Advantages of YOLO

Several factors account for the success of YOLO for object detection:

- **Speed:** YOLO is special because it is unmatched in terms of speed as it bypasses complex processing pipelines. YOLO has the capacity to process images at an incredible 45 frames per second (FPS). To top that, YOLO boasts more than twice the mean Average Precision (mAP) of other real-time systems, hence being an excellent choice for real-time purposes.

- **High Detection Accuracy :**YOLO is better than other state-of-the-art detection models in terms of detection accuracy, with practically no background errors.

- **Better Generalization:**Especially in later versions, YOLO possesses even better generalization capabilities for new domains.

- **Open-source:**The open-source aspect of YOLO has enabled the community to keep modifying the model incrementally.

3.6.2 Achitecture of YOLO

YOLO architecture is similar to GoogleNet. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.

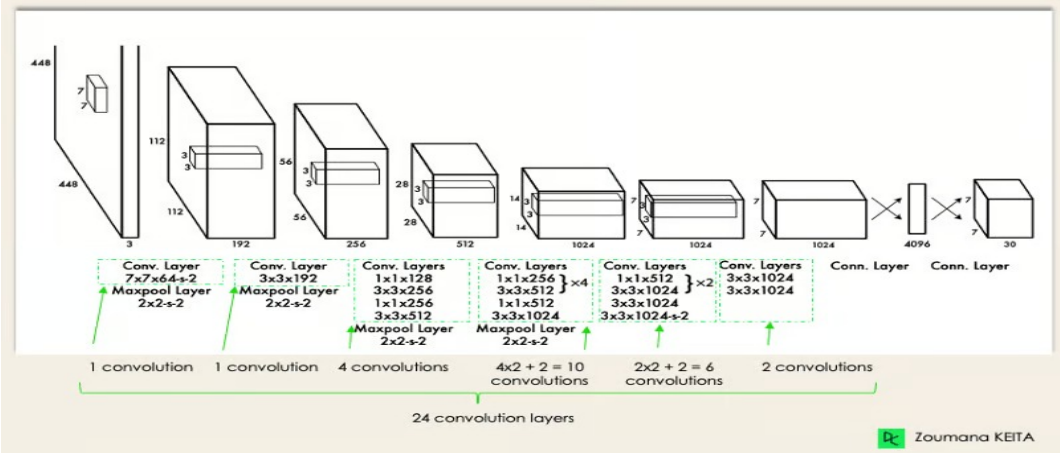

Figure 3.13: YOLO Architecture

The architecture operates as follows:

- The input image is first resized to 640×640 pixels and then fed into the convolutional network.
- First, a 1×1 convolution is employed to reduce the number of channels, followed by a 3×3 convolution to produce a feature map with depth.
- ReLU activation function is utilized everywhere in the network except for the last layer, where linear activation is employed.
- To improve generalization and reduce overfitting, techniques like batch normalization and dropout are employed.

3.7 Related Works

Numerous research efforts have been conducted on animal detection, particularly birds, in airport environments using artificial intelligence, with the aim of preventing wildlife hazards

- **Use of YOLO Models for Bird Detection:** Several studies explore the application of the YOLO (You Only Look Once) model family for bird detection. Ummah et al. [15] developed a model using YOLOv4 for bird detection in airports, achieving a mean Average Precision (mAP) of 71.89%. Their work also compares object detection with motion detection, concluding that object detection performs better under various environmental conditions such as varied lighting, fog, and different distances.se focused on improving the detection of small birds in flight in airport surveillance by introducing channel attention mechanisms into YOLOv5. Their enhanced model, SE-YOLOv5, demonstrates improved detection accuracy and recall rate on a dataset measured at an airport.

The limitations of Ummah et al.'s work primarily concern the nature of the dataset used, the difficulty in accurately detecting small objects, the necessary trade-off to avoid overfitting, and the lack of bird species classification. The authors themselves emphasize the need to improve the dataset and expand the system's capabilities for better performance and more practical application.

- **Advanced Wildlife Hazard Management (AAWM) Systems:** Khan and Shin[3] propose an Advanced Wildlife Hazard Management (AAWM) system that utilizes drone swarms (UAVs) equipped with depth cameras and deep learning models such as Faster R-CNN and YOLOv9 for wildlife detection near airports. Their study indicates that the Faster R-CNN model achieved the best performance, reaching accuracies of 94.34% and 86.13% with IOU thresholds of 0.3 and 0.5 respectively. After detecting birds, the system uses an acoustic deterrent at frequencies of 20 to 25 kHz to disperse them without causing harm. While Khan and Shin's study offers an innovative and promising approach, practical limitations related to drone technology, real-time processing, the effectiveness of deterrents, and system integration may exist and would need to be addressed during actual implementation.

- **Deep Learning Frameworks for Real-time Detection:** Alzadjali et al. [16] developed a deep learning framework for real-time bird detection with the aim of reducing bird strike incidents. Their innovative model is a Spatio-Temporal Convolutional Neural Network (ST-CNN) that integrates spatial attention structures and dynamic temporal processing to accurately recognize birds in flight. The model, composed of a Spatial-Aware Convolutional Network (SACN) and an Attention-Based Temporal Analysis Network (ATAN), demonstrated superior performance in terms of accuracy and real-time response compared to existing bird detection systems. A comparison with other meta-architectures such as Faster R-CNN, R-FCN, Retina Net, SSD, YOLO v4, and YOLO v5 confirmed the better efficiency of the proposed model. The study also addresses data preprocessing, augmentation, training, and model validation.

These related works do not consider the detection of mammals, especially when it comes to feral animals, but they highlight the growing interest in the application of artificial intelligence, particularly deep learning techniques and object detection models like YOLO and Faster R-CNN, to enhance airport safety by detecting and managing wildlife hazards. The research explores different model architectures, the integration of spatial and temporal data, the use of drones and acoustic deterrent systems, as well as optimization for real-time detection in varied environmental conditions.

This chapter laid a solid groundwork for our proposed AI-driven animal detection system specifically designed for airport scenarios. It explored how models such as Faster R-CNN and YOLOv11n can be employed for real-time detection of wildlife. By studying the strengths and weaknesses of existing approaches, our approach will be centered on achieving appropriate balance between detection rate and accuracy to deliver an honest-to-goodness and cost-effective system for wildlife risk mitigation. The next chapter will discuss the specifics of the system and training parameters.

Bibliography

Bibliography

[1] Federal Aviation Administration, "National wildlife strike database," 2023.

[2] E. C. Cleary and R. A. Dolbeer, *Wildlife Hazard Management at Airports: A Manual for Airport Personnel*. Washington, DC: Federal Aviation Administration and U.S. Department of Agriculture, 2005.

[3] F. A. Khan and S. Y. Shin, "Advanced avian wildlife management (aawm) system through deep learning based uav swarm and sonic deterrents for airport safety," *2024 KICS*, June 2024.

[4] R. A. Dolbeer and S. E. Wright, "Wildlife strikes to civil aircraft in the united states, 1990–2007," Tech. Rep. Serial Report Number 14, Federal Aviation Administration, Washington, DC, 2008.

[5] International Air Transport Association, "Annual report on global air traffic," 2023.

[6] Animal and Plant Health Inspection Service, "Wildlife hazards at airports," 2023.

[7] International Civil Aviation Organization, "Annex 14, volume i: Aerodromes," 2023.

[8] Federal Aviation Administration, "14 cfr part 139.337: Wildlife hazard management," 2023.

[9] U.S. Fish and Wildlife Service, "Migratory bird treaty act of 1918," 2023.

[10] I. C. A. Organization, *Airport Services Manual, Part 3: Wildlife Control and Reduction*. Montreal, Canada: ICAO, 2012.

[11] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, *Advances in Deep Learning*. Springer, 2019.

[12] A. K. Das, J. Nayak, B. Naik, S. K. Pati, and D. P. Editors, *Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2019*, vol. 999. Springer, 2019.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[14] Joseph Redmon and Ali Farhadi , "Yolo: Une brève histoire," 2025.

[15] U. Khairul *et al.*, "Bird detection system design at the airport using artificial intelligence," *International Journal of Aviation Science and Engineering*, vol. 4, pp. 59–67, 2022.

[16] A. Alzadjali *et al.*, "Deep learning frameworks for real-time detection," *Sensors*, vol. 24, no. 17, p. 5455, 2024.