Aristotle University of Thessaloniki
Faculty of Engineering
School of Electrical and Computer Engineering
Intelligent Systems and Software Engineering Laboratory

Implementation of a platform for assessing indoor spaces regarding their friendliness to people with dementia.

Postgraduate Thesis
of

Triantafyllidis Dimitris
(AEM 494)

Supervising: Prof. Andreas Symeonidis
Dr. Tsardoulias Emmanouil

April 25, 2024

Acknowledgements

I would like to express my gratitude to Prof. Andreas Symeonidis for granting me the opportunity to undertake my thesis within the Intelligent Systems and Software Engineering Labgroup (ISSEL) at Aristotle University of Thessaloniki, in the Department of Electrical and Computer Engineering.

I'd also like to thank Dr. Tsardoulias Emmanouil for his constant support and guidance during my thesis work. As the technical leader of ISSEL, he has been incredibly helpful, always ready to share his expertise, knowledge and advice. His support played an important role in helping me to overcome the challenges of my thesis and reach its successful completion.

I'm deeply grateful to my parents, Charalampos and Athanasia, for their unconditional support, love, and the sacrifices they've made for me. Their belief in my abilities has always been a source of motivation, pushing me to strive for my best. A special shout-out to my brother, Lazaros, for his constant encouragement and help in every aspect of my life, beyond just my thesis work. His support has been a key factor in my achievements and happiness.

Finally, I wish to thank my relatives, friends, and acquaintances for their contributions to my personal and professional growth.

Title

Implementation of a platform for assessing indoor spaces regarding their friendliness to people with dementia

Abstract

This thesis presents an innovative tool specifically designed to assess the dementia-friendliness of indoor spaces. With the rise in dementia cases globally, the need to adapt environments to the unique needs of those affected is becoming increasingly important. This research focuses on evaluating how friendly and suitable these spaces are for individuals with dementia, with a particular emphasis on flooring.

The developed tool leverages the power of deep learning and machine learning to analyze images of indoor spaces, identifying objects within these images to evaluate their suitability for dementia-friendly environments. The tool currently focuses on evaluating rugs, equipped with a easy to navigate and effective web application that allows caregivers, designers (or other users) to easily upload photos for assessment. This specialized approach ensures that users can quickly understand how these floor coverings align with dementia-friendly design principles.

The tool is user-friendly and efficient, designed for quick assessment of indoor areas. Rather than suggesting changes, it rates how dementia-friendly these spaces are, providing valuable insights for caregivers and designers. This feature is essential for public places such as care homes, private residences or even hotels that aim to be dementia-friently, where creating safe, understandable, and comfortable environments for people with dementia is crucial.

Overall, this thesis merges technology with the principles of interior design in the context of dementia care. It offers a significant contribution to the field, providing a practical and innovative approach to assess and understand dementia-friendly environments.

Keywords: Dementia, Interior Design, Deep Learning, Machine Learning, Indoor Space Assessment, Supportive Living Spaces, Assessment Tool

Dimitrios Triantafyllidis
Electrical & Computer Engineering Department,
Aristotle University of Thessaloniki, Greece

March 2024

Contents

List of Figures

Acronyms

Below there are listed some of the most commonly used acronyms in the present thesis:

DL	$\longrightarrow$	Deep Learning
AI	$\longrightarrow$	Artificial Intelligence
ML	$\longrightarrow$	Machine Learning
YOLO	$\longrightarrow$	You Only Look Once
ResNet	$\longrightarrow$	Residual Network
MLP	$\longrightarrow$	Multilayer Perceptron
CNNs	$\longrightarrow$	Convolutional Neural Networks
FC	$\longrightarrow$	Fully Connected Layers
BE	$\longrightarrow$	Backend
FE	$\longrightarrow$	Frontend
IBE	$\longrightarrow$	Indoor built environment
QoL	$\longrightarrow$	Quality of life
C & A	$\longrightarrow$	Care Attention

1 Introduction

1.1 Problem Overview

Dementia, a complex neurological disorder, affects millions worldwide and presents unique challenges in the design of indoor spaces. As stated in [1] the care environment plays a crucial role to support people with dementia in their daily functioning and well-being. Also [2] suggests that using principles from psychology to design care homes can help older people, especially those with Alzheimer's, navigate better and feel more independent. The World Health Organization (WHO) emphasizes the growing prevalence of dementia, with over 55 million people currently affected and projections indicating a rise to 139 million by 2050. This increasing trend underscores the critical need for environments that cater to the specific needs of individuals with dementia [3].

Building on this need, recent research has highlighted how making specific changes to the environment can greatly improve life for people with dementia [4]. For example, a study conducted in care homes in Hong Kong found that factors like better lighting, controlled temperatures, and easy-to-navigate layouts had a positive effect on the residents' physical health, mental well-being, and social interactions [5]). Moreover, there is a general agreement among experts that people with dementia benefit from living in smaller, more home-like care settings, linked to better mental health care, increased social activity, and a reduction in behavioral problems (Emerald Insight, 2019).

These findings underscore the critical role of environmental design elements, such as lighting and spatial layouts, in enhancing the quality of life for those with dementia. Additionally, among these elements, flooring emerges as a particularly significant factor [6]. The choice of flooring can notably influence navigation, safety, and comfort. For individuals with dementia, certain types of flooring, especially those with complex patterns or high contrasts, can lead to confusion and disorien-

tation [7]. Research indicates that plain surfaces are preferable, as patterns can create visual disturbances to the individual with dementia that lead to difficulties in perception.

Given these considerations, the escalating demographic challenge necessitates an innovative approach in assessing and adapting indoor spaces to be more dementia-friendly. The development of a systematic assessment tool is thus crucial. Such a tool would not only evaluate and guide modifications in key environmental factors, including flooring, spatial layout, signage clarity, and lighting quality, but also aim to facilitate better navigation, enhance safety, and reduce stress and confusion for individuals with dementia. By creating an environment that supports independence and cognitive orientation, this tool can significantly improve their daily living experience.

The following section delves into the specific objectives and anticipated impact of the proposed assessment system. It outlines the potential to transform indoor environments into supportive spaces that enhance the well-being and autonomy of individuals living with dementia.

1.2 Purpose of Thesis

The main goal of this thesis is to create and deploy a tool (website) for assessing the dementia-friendliness of indoor environments, with an initial focus on flooring. This effort responds to the rising global incidence of dementia and the associated need for spaces that accommodate the unique needs of affected individuals.

Utilizing deep learning techniques, the developed tool enables users to upload photos of various indoor spaces. It is designed to assess these environments for dementia friendliness by analyzing images, identifying objects, and evaluating their contribution to the well-being of individuals with dementia. The initial concentration on flooring stems from its significant impact on the safety, comfort, and navigability for people with dementia.

A notable feature of this tool is its configurability and potential capability for extension of the object categories it can detect in an image, to evaluate if they should be placed in dementia friendly environments.

This research is expected to deepen the understanding of dementia-friendly design, particularly in flooring, and to develop an assessment tool beneficial for caregivers, designers, and other users. The ultimate goal is to enhance the living environments of individuals with dementia, making them safer, more comprehensible, and comfortable.

1.3 Thesis Structure

This section offers an epigrammatic overview of the thesis structure, providing a navigation through the chapters to highlight the key components of the thesis.

1. **Chapter 1: Introduction** This chapter defines the problem: the necessity for dementia-friendly indoor environments. It not only defines the problem but

also proposes a novel tool designed to detect such spaces, thereby facilitating progress towards creating more dementia-accommodating environments in public environments. The structure of the thesis is also briefly described, providing readers with an overview of the thesis.

2. **Chapter 2: Review of Literature** This chapter examines prior research focused on evaluating indoor environments for individuals with dementia. The chapter discusses how various aspects of public environments can impact individuals with dementia—drawing from psychological insights into what can disturb, annoy, or confuse them. These insights are then applied to enhance the design of the proposed system, ensuring it effectively identifies dementia-friendly environments.

3. **Chapter 3: Theoretical Background & Basic Principles** This chapter provides the theoretical background of the research, starting with an explanation of dementia and its impact on perception. The chapter also introduces artificial intelligence, neural networks, and object detection technologies, including a discussion on Convolutional Neural Networks (CNNs) and specific models like YOLOv7 and ResNet, establishing the scientific basis for the methodology.

4. **Chapter 4: Methodology** In this chapter the system design and architecture of the tool is described, including the implementation details of various components that is comprised. The approaches to containerizing and deploying the tool are explained, offering insights into how future development and maintenance can be streamlined.

5. **Chapter 5: Experiments & Results** This chapter presents the experimental setup, including the selection of test images and the configuration of detection and classification models. It then details the results of object detection, assessing the accuracy, performance, and user interaction. Through experiments and examples, the tool's capability to assess indoor spaces for dementia-friendliness is showcased. Additionally, screenshots from the tool (web application) are provided to illustrate its functionality and the results it generates.

6. **Chapter 6: Conclusions and Future Work** The final chapter summarizes the findings of the thesis, acknowledging its limitations and challenges. It outlines future directions, suggesting areas for business improvements, user interface and experience enhancements, and the potential expansion of object categories within the tool.

2

Review of Literature

2.1 Previous Studies on Indoor Space Assessment For People with Dementia

As the global demographic ages, designing environments that support the well-being of individuals with dementia becomes increasingly critical. This section examines important studies, such as the research conducted by Leung et al. [8] that highlights the critical role of the indoor built environment (IBE) in enhancing the quality of life (QoL) for demented elderly residents in care and attention (C & A) homes. Three essential components of the IBE are being mentioned, each incorporating a series of specific factors, labeled (F1- F12), that are essential in optimizing the residents' QoL. These components include:: space management, building services, and supporting facilities. Space management, addressing elements like distances (F1) and privacy (F2), is key as it helps prevent disorientation among the elderly. Building services, encompassing lighting (F3), ventilation (F4), temperature (F5), lifts (F6), and water supply (F7), are fundamental in maintaining the residents' physical comfort and safety. Supporting facilities, including handrails (F8), signage (F9), finishes (F10), furniture (F11), and recreational facilities (F12), contribute significantly to the residents' independence and psychological well-being.

Delving deeper, the research analyzed how these IBE elements influence the QoL of the demented elderly, segmented into six major factors (Q1 - Q6): physical health (Q1), psychological health (Q2), independence (Q3), activities of daily living (Q4), social relationships (Q5), and cognitive functioning (Q6). The comprehensive study was conducted across eight C & A homes and multiple research methods were used, involving interviews with 18 experienced caregivers and a survey targeting individuals with mild to moderate dementia. This mixed-method approach offered valuable insights into how the Indoor Built Environment (IBE) significantly improves the QoL for the demented elderly in these settings. Also to strengthen the validity of the

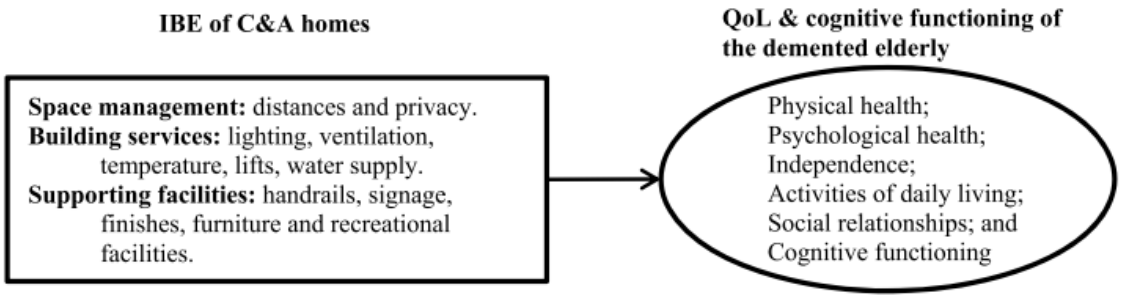

Figure 2.1: Conceptual IBE-QoL model from the study [8]

results they used reliability tests (Cronbach's alpha) which were applied to validate the QoL and IBE factors, followed by correlation and multiple regression analyses to explore their relationships 2.2. The qualitative data collected from interviews with caregivers, were analyzed through content analysis to add depth to the findings. This approach helped identify key IBE factors impacting the elderly's QoL. Their results were the following [8]:

1. **Space Management (F1-F2):**
    - **Distance (F1), Privacy (F2)**
        - Suprisingly neither of the space management components significantly affected the **QoL** of the demented elderly;
2. **Building Services (F3-F7):**
    - **Temperature (F5):**
        - Inappropriate temperature limits activities of daily life (**Q4**).
        - Essential for comfort and participation in daily activities.
    - **Lifts (F6):**
        - Enhance the independence (**Q3**) of the demented elderly.
        - Facilitate mobility and access within the facility.
3. **Supporting Facilities (F8-F12):**
    - **Signage (F9):**
        - Positively influences social relationships (**Q5**) and cognitive functioning (**Q6**).
        - Facilitates elderly orientation and location identification.
    - **Finishes (F10):**
        - Significantly affects physical health (**Q1**), activities of daily life (**Q4**), social relationships (**Q5**), and cognitive functioning (**Q6**).
        - Color contrasts assist in area identification and improve comfort.
    - **Furniture (F11):**

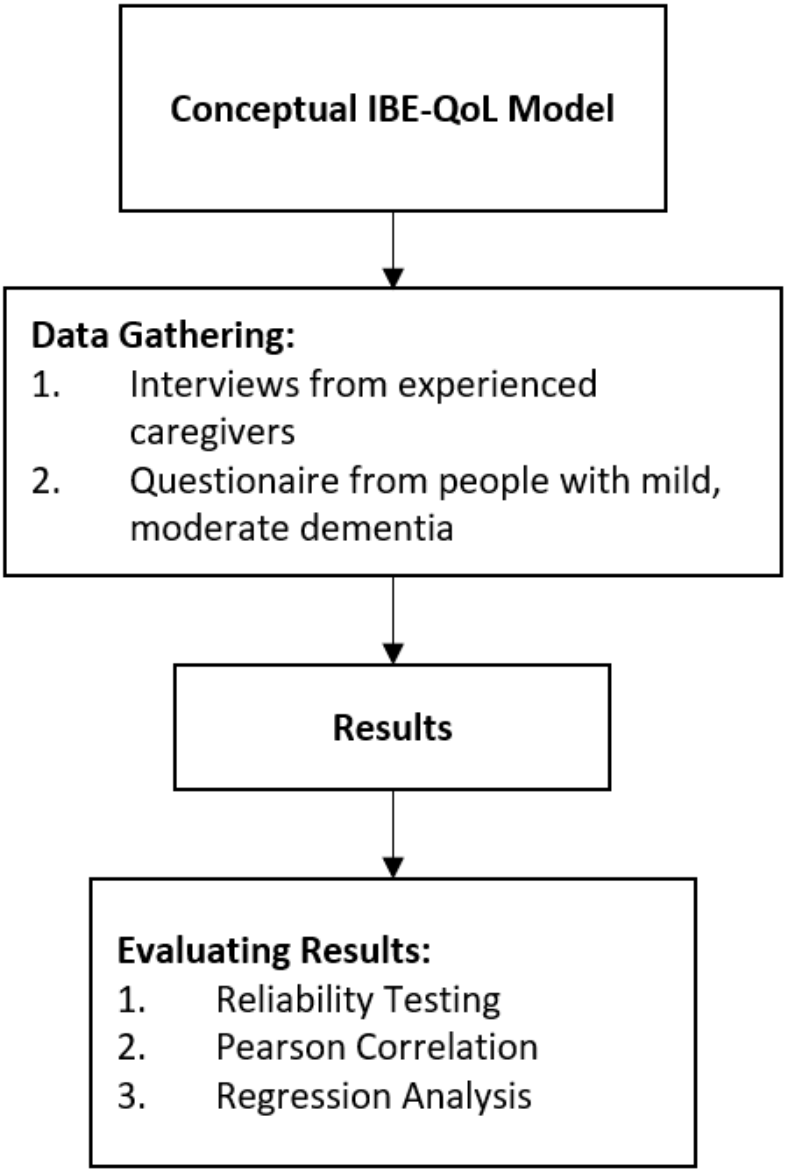

Figure 2.2: Methological framework of [8]

- Dissatisfaction linked to reduced physical health (**Q1**) and cognitive functioning (**Q6**).
- Appropriate furniture aids in spatial orientation.

As presented in [9], the authors concentrate on design interventions in long-term care facilities to enhance social interaction, a key factor in residents' quality of life. The study, synthesizing research from various post-2000 sources, identifies essential spatial design elements for fostering social engagement: the physical environment and setting, accessibility and layout, social environment and network, and the staff-resident ratio and care philosophy. These factors are highlighted as crucial in creating positive social interactions within care settings. The paper provides guidelines on designing environments to boost the well-being and quality of life for caregivers, staff, and residents.

- **Open & Clear Layout:** Design spaces that are easy to navigate and encourage socializing.
- **Small-Scale Facilities:** Opt for mid-size or smaller facilities to boost social interaction in a home-like setting.
- **Enhanced Mealtimes:** Focus on dining experiences to improve mood, nutrition, and social bonds.
- **Comforting Ambiance:** Use lighting, music, and aromas to create a soothing indoor environment.
- **Therapeutic Gardens:** Provide safe, sensory-rich outdoor areas for relaxation and socializing.
- **Natural Settings:** Integrate green spaces and nature-based activities for engagement and well-being.
- **Homelike Decor:** Personalize spaces to foster connections among residents, families, and staff.
- **Easy Navigation:** Use distinct decor and clear signs to help with orientation and independence.
- **Privacy & Family Inclusion:** Private rooms and family-friendly areas support comfort and social interaction.
- **Social Seating Arrangements:** Arrange furniture to encourage conversation and avoid isolating seating styles.
- **Empower Residents Through Autonomy:** Value residents' choices in design decisions to enhance social interaction, quality of life, and well-being. Include them in decisions about landscaping, artwork placement, indoor plants, home décor, and furniture layout in common areas and bedrooms.
- **Provide Ample Staff–Resident Ratio:** A high staff-to-resident ratio is crucial for social interaction and is often considered more important than the physical environment. Quality interactions between staff and residents are key to promoting the psychological and psychosocial well-being of people with dementia.

Staff presence significantly influences the overall psychosocial climate and can impact residents' well-being positively or negatively.

In their insightful work, [10], the authors explore how the built environment affects the well-being of people with dementia. They suggest practical design approaches focused on three main ideas: keeping cognitive demands manageable, ensuring clear sequence in environments, and maintaining suitable levels of stimulation. These concepts are part of a larger framework that also considers crucial factors like social support and medical care. The study reveals that addressing these areas together can greatly improve a person's quality of life. Notably, it features a model (see Fig. 2.3) showing the typical decline in quality of life for dementia patients, indicating that well-planned changes in the **environment**, **social support**, and social **health care** can make a positive difference. This approach highlights the importance of a coordinated effort in these key areas to enhance life for those with dementia.

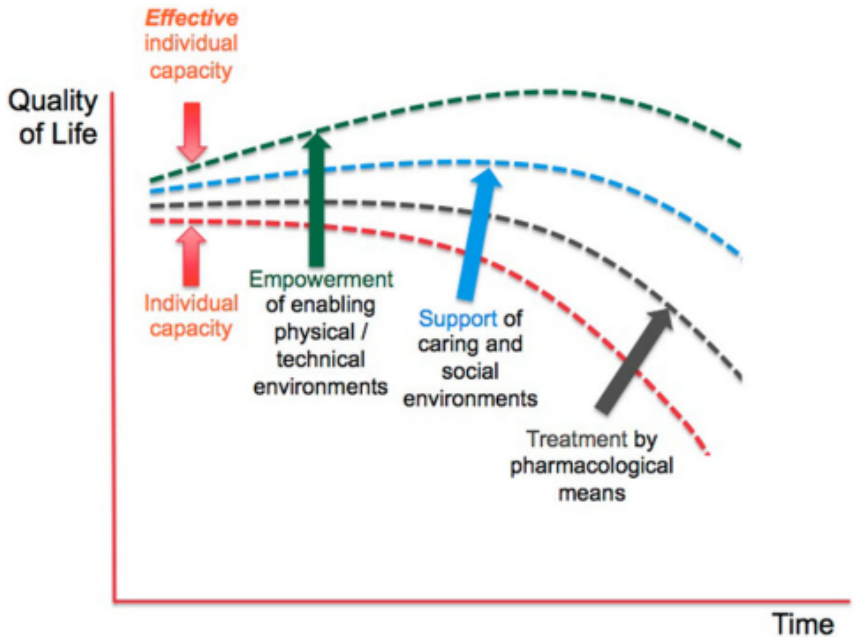

Figure 2.3: Qol to time chart [10]

Kelly et. al [5], investigate how well care homes support people with dementia, using two tools – the Design Audit Tool and the Environmental Audit Tool. Their study, conducted in 30 care homes, aims to validate these tools by assessing their reliability in measuring the suitability of care home environments for individuals with dementia. The researchers identify six crucial themes that are instrumental in creating a dementia-friendly environment: **1) Wayfinding**, emphasizing the importance of clear navigation aids within care homes; **2) Use of Colour**, focusing on how contrasting colors can aid residents with visual processing; **3) Individualization of Space**, highlighting the need for personal and homely spaces; **4) Outside Space**, underscoring the importance of accessible and stimulating outdoor areas; **5) Lighting**, recognizing the need for adequate and appropriate lighting to compensate for sensory changes in dementia; and **6) Opportunities for Engagement with the Environment**, stressing the importance of creating engaging and stimulating spaces.

Their research concludes with valuable insights, revealing significant variations in the adherence to these dementia-friendly principles across different care homes. This study not only sheds light on the current state of care home design for individuals with dementia but also provides a framework for assessing and improving these environments.

1. Signage
    - Provide clear signage at key points, including main entrances, toilets, lounges, and reception desks.
    - Use a combination of text and pictorial signs for easy identification of different rooms.
    - Employ upper and lower case lettering in large, simple fonts with recognizable symbols. Avoid abstract or joke signage.
    - Ensure high contrast between sign wording and background.
    - Place signs at eye level and ensure visibility from a wheelchair, if applicable.
2. Visibility and Lighting
    - Keep curtains open during the day and remove any obstructions that block natural light.
    - Consider the use of automatic lighting, particularly in restrooms.
    - Outline steps and stairs for better visibility.
    - Use contrasting colors for door handles and doors, and ensure doors stand out against adjacent walls.
    - Avoid bright light pools, glare, and deep shadows. Ensure light switches are easily accessible.
3. Toilets
    - Provide unisex accessible toilets and, ideally, Changing Places toilets.
    - Ensure cubicle doors are clearly visible with appropriate handles.
    - Use 'Way out' signs inside toilet doors.
    - Label taps and provide instructions for sensor taps, flushes, and hand dryers.
    - Offer contrasting toilet seats and handrails.
4. Seating and Flooring
    - Provide high-backed chairs with armrests in waiting and quiet areas.
    - Ensure seating contrasts with the floor and surroundings. Avoid abstract designs.
    - Steer clear of dark rugs, shiny or reflective flooring, and bold patterns that can cause confusion.

5. Furnishings and Facilities
    - Avoid shiny, reflective surfaces and confusing patterns in furnishings and window coverings.
    - Position mirrors carefully and provide coverings if necessary.
    - Provide spare linen in guest rooms, simple bedside clocks, and large clocks in key areas.
    - Implement a simple system for contacting staff for assistance.
    - Avoid rugs and cluttered furniture that could create trip hazards. Ensure ample space for movement.
    - Clearly label items for making drinks, like tea and coffee.
    - Consider providing accessibility products like wheelchairs, large-button phones, and magnifying glasses.
6. Noise
    - Reduce background noise from various sources to improve concentration and communication.
    - Use carpets, cushions, and curtains to absorb background noise.
    - Promote 'slow' or 'relaxed' times with reduced noise levels and provide a quiet room.
    - Ensure regular servicing and proper advertising of hearing loops.

In contrast to existing approaches, this thesis introduces a tool that leverages deep learning and machine learning techniques for the analysis of indoor spaces, aiming specifically to improve the lives of people with dementia. Unlike previous studies that primarily focus on theoretical frameworks or manual assessments, the developed tool automates the evaluation process of indoor environments. It offers a platform, allowing caregivers and designers to upload images for instant assessment. This innovative approach not only simplifies the process of assessing whether environments align with dementia-friendly design principles but also provides statistics regarding the dementia-friendliness level of the environments depicted in the uploaded images.

3

Theoretical Background & Basic Principles

3.1 Dementia and Its Impact on Perception

3.1.1 Definition of dementia

Dementia is a broad term that refers to a variety of progressive neurological disorders. These disorders significantly impair a person's cognitive functioning, hindering their ability to perform everyday tasks and affecting their daily life. It is characterized by a gradual decline in memory, thinking, comprehension, arithmetic, language, learning, and judgment abilities [11]. Impairments in cognitive function are often accompanied by, and sometimes preceded by, impairments in emotional regulation, social behavior, or motivation [12]. Although aging is a significant risk factor, dementia is not a normal part of aging [13].

There are many different types of dementia, but the most common type is Alzheimer's disease. It should be noted that dementia is progressive, which means it begins with mild symptoms that get worse over time. The progression of dementia will vary between individuals and people will experience it in different ways [14] [15] [16]. Since no cure is currently available, prevention is crucial.

3.1.2 Symptoms of Dementia

The most well-known symptoms of dementia are the following: [16]

- **Memory loss** – For example, a person may:
  - Struggle to recall recent happenings, yet clearly remember events from years ago.

  - Repeatedly saying the same things or asking the same questions over and over.

- **Difficulty thinking things through and planning** – For example a person might notice things like:
  - Finding it hard to stay focused, follow step-by-step processes, grasp new ideas, or tackle problems.
  - Having trouble with everyday tasks, for example, cooking with a recipe or using bank cards.
- **Problems communicating** – For example, a person may:
  - Experience trouble in finding the right words to express thoughts.
  - Struggle to keep up with conversations or misunderstanding what others are saying.
- **Being confused about time or place** – For example, a person may:
  - Lose track of what time, date, or season it is.
  - Feeling disoriented, even in familiar places.
- **Problems with sight and visual perception** – For example, a person may:
  - Have difficulty judging distances, such as on stairs.
  - Misinterpret patterns or reflections in mirrors.
- **Mood changes or difficulties controlling emotions** – For example a person may:
  - Experience unusual or sudden feelings of sadness, fear, anger, or getting upset easily.
  - Show a lack of interest in previously enjoyed activities, or becoming more withdrawn.
  - Show a noticeable decrease in self-confidence.

3.1.3 Improving the physical environment

In individuals with dementia, environmental factors play a significant role in their ability to navigate and interact with their surroundings. Even small modifications in the layout, like rearranging furniture or enhancing signage, can substantially reduce stress and confusion. These enhancements don't necessarily require extensive refurbishments or high costs. When planning renovations or redecorating, it's beneficial to consider cost-effective adaptations that could make the environment more dementia-friendly. Such thoughtful changes can create a more supportive and navigable space, contributing positively to the well-being of those with dementia [16].

3.2 Introduction to artificial intelligence

Artificial Intelligence (AI) represents a vast field in research and technology, where machines exhibit cognitive abilities similar to humans, such as learning behaviors, predictive interactions with the environment, data analysis, and decision-making. AI mimics intelligent behaviors typically associated with humans, drawing inspiration from computer science, mathematics, and statistics. A subset of AI, **Machine Learning** (ML), focuses on enabling computers to learn tasks without being explicitly programmed. ML is grounded in the idea of creating algorithms that learn from data and make predictions or decisions. There are **three main categories within ML**:

1. **Supervised Learning**: In this approach, the computer is given sets of input and corresponding output examples, already labeled. The aim is to derive a broad rule that connects the input with the correct output.
2. **Unsupervised Learning**: This method involves working with unlabeled data, requiring the ML algorithm to independently identify patterns and structures within the input data.
3. **Reinforcement Learning**: A computer program operates within a dynamic environment where it aims to achieve a specific goal, like controlling a vehicle or competing in a game. Throughout its operation, it receives feedback similar to rewards. The program's objective is to maximize these rewards as it maneuvers through its set of challenges.

Deep Learning, a methodology within ML, employs artificial neural networks with multiple layers for training models. It relies on representations of training data, as opposed to algorithms designed for specific tasks. Deep learning can be supervised, unsupervised, or reinforcement-based, and is used in various scientific fields such as computer vision, speech recognition, natural language processing, sound recognition, social network filtering, machine translation, bioinformatics, drug design, and more. In Fig. 3.1 the relationship between the above field can be depicted: AI is the most general category, containing ML, and within ML, there is Deep Learning, indicating how each area is a more specific part of the larger field.

In this thesis, **Deep Learning** and **neural networks** are utilized and will be explained further in the next subsection. The focus will be on understanding how they function and their relevance to the topics covered in this study.

3.3 Neural Networks and Biological Inspiration

The concept of neural networks in computing is based on the human brain's neural network. In biology, a neural network includes:

- **Neurons:** Cells that handle information processing and communication. In essence, they are used for handling inputs and generating outputs in the brain.
- **Dendrites:** Structures that receive input signals from other neurons,
- **Axons:** Connects transmitting (output) signals to other neurons.

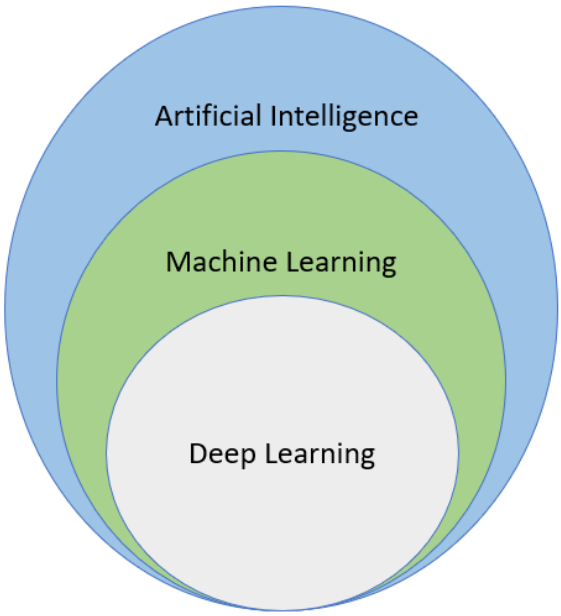

Figure 3.1: Venn diagram capturing the relationship between, AI, ML and DL.

- **Synapses:** The junction points where neurons connect in order to communicate and transfer signals (inputs and outputs) between neurons.

Artificial neural networks, used in computing, mirror these biological components:

1. **Artificial Neurons (Nodes):** Just like biological neurons, artificial neurons (or nodes) are fundamental units for processing information. In the human brain, neurons process and transmit information through electrical and chemical signals. In ANNs, artificial neurons receive, process, and transmit information in the form of numerical values and calculations.
2. **Weights and Biases:** Weights in ANNs are similar to the strength of synaptic connections in biological neurons. Just as the strength of a synaptic connection determines how strongly one neuron will affect another, weights in ANNs determine the influence of one artificial neuron's output on another's input.
3. **Activation Functions:** These functions in a node bring in non-linearity, essential for complex pattern learning, much like the role of axons and synapses in the brain.

In terms of mathematics in artificial networks:

- Inputs X in a neural network are analogous to the signals received by dendrites in a biological neuron. These inputs can be represented as a vector $x = [x_0, x_1, ..., x_{N-1}]$, where each element x_i represents a different input value.
- Weights W correspond to the influence each input signal has, similar to how a neuron modulates signal strength. The weights are represented as $W = [w_0, w_1, ..., w_{N-1}]$, with each weight w_i associated with the corresponding input x_i.
- The bias b acts as an offset, similar to how a neuron integrates signals. It is a

single value added to the weighted sum of inputs.

- The weighted sum of inputs, including the bias, is given by:

$$Z = \sum_{i=0}^{N-1} w_i \cdot x_i + b \tag{3.1}$$

This equation accumulates the weighted inputs and adjusts them with the bias.

- The activation function f processes this sum Z to produce an output a, analogous to how an axon transmits signals in a biological neuron. The output of a neuron is given by:

$$a = f(Z) = f\left(\sum_{i=0}^{N-1} w_i \cdot x_i + b\right) \tag{3.2}$$

This function introduces non-linearity, allowing the neural network to learn and model complex patterns and relationships.

In artificial neural networks, the role of an activation function, also known as a transfer function, is crucial in determining a neuron's output based on its input or set of inputs. These functions can be either linear or non-linear. The incorporation of non-linear activation functions is essential, as they enable the network to model complex non-linear relationships between inputs and outputs. The selection of an appropriate activation function is largely influenced by the specific problem the neural network aims to solve, as well as its overall structure and architecture. While certain activation functions are more commonly used due to their effectiveness in various scenarios, this does not imply that they are universally applicable to all contexts. The most common type of activation function is the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.3}$$

Alternatively, the sigmoid activation function can expressed in discrete form as

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{3.4}$$

This is only known as the step function. Using step function as an activation function, it becomes clearer how the most basic form of the simplest **artificial neural network** (ANN), the perceptron (Fig. 3.2), computes its output. The perceptron, created by Frank Rosenblatt in the 1950s and 1960s [17], can be characterized as the simplest type of a **feedforward network**. In a feedforward network, data flows in a single direction: it starts at the input and ends at the output, without any looping back. The perceptron operates in a clear and straightforward manner. It starts by taking in several inputs (e.g X vector), each assigned a specific weight (e.g W vector) to signify its relative importance. The perceptron then computes the sum of these weighted inputs (e.g Z). This sum is processed by an activation function, which in the initial models of perceptrons was typically a step function (e.g $a = f(Z) = f\left(\sum_{i=0}^{N-1} w_i \cdot x_i + b\right)$ where f the activation function). The role of this

activation function is crucial – it determines the perceptron's output based on the computed sum. If this sum reaches or exceeds a certain threshold, the perceptron activates and produces an output of 1. Otherwise, it outputs 0. By utilizing the step function as its activation function, the perceptron effectively acts as a binary classifier, categorizing input data into one of two distinct classes or categories.

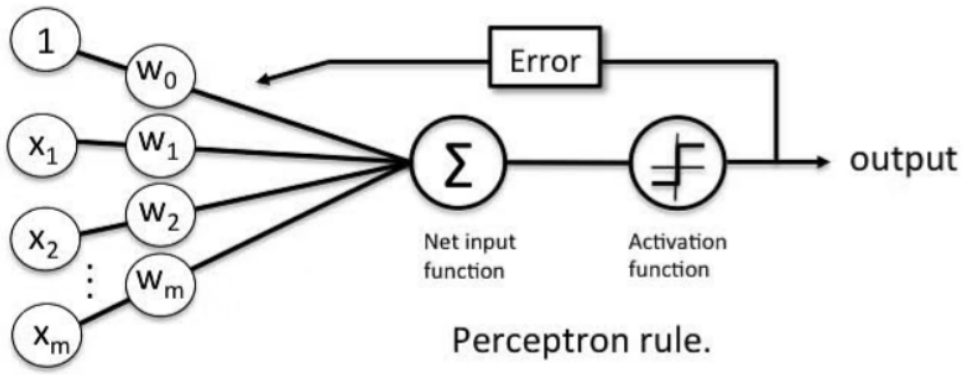

Figure 3.2: A schematic diagram of a perceptron featuring four input neurons and one output neuron.

The perceptron is a foundational element of neural networks, but it's limited in its capability to solve complex problems. To overcome these limitations, the field of neural network research has evolved towards more advanced architectures, notably the Multilayer Perceptron (MLP).

3.3.1 Multilayer Perceptron- (MLP)

The Multilayer Perceptron (MLP) [18] is a type of feedforward neural network that introduces hidden layers between the input and output layers. This architecture marks a significant advancement from the single-layer perceptron model. In an MLP, each layer of nodes, or neurons, feeds forward to the next layer. The introduction of multiple hidden layers allows the network to capture and model complex, non-linear relationships in data, which was a limitation in single-layer perceptrons. The structure of an MLP can be depicted in (Fig. 3.3) and it consists of the following layers:

1. **Input Layer**: This layer receives the input signals.
2. **Hidden Layers**: These layers perform computations on the inputs received and pass their output to the next layer. The more hidden layers (and more nodes in each layer) a network has, the more complex patterns it can learn. However, more layers also mean more computational complexity and a risk of overfitting. The simplest example of an multilayer perceptron contains at least on hidden layer.
3. **Output Layer**: Produces the final output of the network.

A Multilayer Perceptron can be used for supervised learning problems and is capable of solving both classification and regression. The ability of an MLP to handle

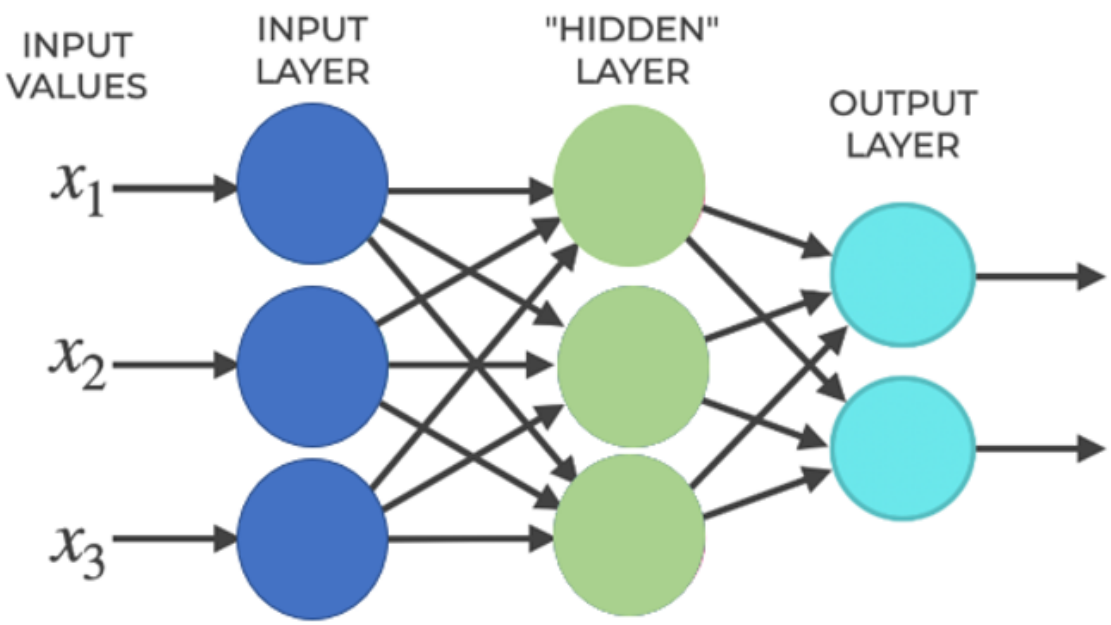

Figure 3.3: The simplest example of an MLP with only one hidden layer.

these types of tasks effectively is influenced by several factors, including the selected activation function. For instance for:

1. **Classification Problems**: The goal in classification is to assign discrete labels to data points (inputs). For binary classification problems (e.g label/class is cat/dog) the output layer typically employs sigmoid activation function. Sigmoid maps the input values between [0, 1]. This output is interpreted as a probability: if the value is above 0.5, the input is more likely to belong to one class (e.g., 'dog'), and if it's below 0.5, it belongs to the other class (e.g., 'cat'). In an MLP used for multi-class classification the output layer typically employs a softmax activation function [19].

2. **Regression Problems:** In regression, the objective is to predict continuous values (like house prices). For regression tasks, the output layer of an MLP often uses linear or identity activation functions. This is because, in regression, a quantitative output is required without the need of transform it to probabilities.

The choice of activation function in the hidden layers can also influence the performance of the MLP. Common activation functions for hidden layers include ReLU (Rectified Linear Unit [20]), tanh (hyperbolic tangent) [21], and sigmoid. These functions introduce non-linearity to the model, enabling it to learn more complex patterns in the data.

3.3.2 Training Process of an MLP

The process initiates with setting up **initial weights and biases** for both the input and the hidden layers. This initial setup is crucial since the choice of the initial learning parametres (weights w and biases b) significantly influences the learning efficiency and overall performance of the network. The next phase is the **forward**

pass, where input data (weights X enters the network and passes through each layer sequentially. As the data travels through each layer, the network computes outputs using its existing weights and biases along with the activation functions (different for each layer). These calculations transform the data progressively as it moves from one layer to the next. The final output is the prediction of the network. To assess the accuracy of the network the **loss function** is employed. Its main purpose is to measure the difference between the network's predictions and the actual correct responses, which is termed as the 'loss'. After the assessment step, **optimization algorithms** are used to reduce the loss by systematically adjusting the weights and biases. To recalibrate the weights and biases using the optimization algorithms, the gradient of the loss function with respect to each weight and bias is required. This step involves a **backward pass**, where the network computes the gradients of the loss function with respect to each weight and bias across all layers and nodes. This phase is commonly referred to as **backpropagation**. In backpropagation, the network assesses how each weight and bias affects the loss function. The process is iterative, continuously aiming to reduce the loss by systematically refining the adjustments made to the weights and biases. Each phase of the process will be analytically explained in the following subsections, offering a comprehensive and clear understanding of each step involved.

3.3.2.1 Forward pass algorithm

The forward pass in a multilayer perceptron (MLP) is the process of calculating the output of the neural network for a given input. This involves computing the weighted sum of inputs (including biases) at each node (neuron) in each layer, applying an activation function to these sums to introduce non-linearity, and then passing the output to the next layer until the final output is produced.

In this section, we examine an example involving a multilayer perceptron (MLP) with l hidden layers. Before proceeding to the calculation of each output, the notation will be explained:

- $W_{ij}^{[l]}$ denotes the weight from node i in layer l-1 to node j in layer l
- $b_j^{[l]}$ denotes the bias term for node j in layer l
- $z_j^{[l]}$ denotes the weighted sum of input to node j in layer l
- $a_j^{[l]} = f(z_j^{[l]})$ denotes the output of the j node in l layer after the activation function f has been applied

The output from each neuron can be calculated from the following function:

$$a_j^{[l]} = f(z_j^{[l]}) = \sum_{j=1}^{J} W_{ij} * a_i^{[l-1]} + b_j^{[l]}, \tag{3.5}$$

where J the number of nodes in layer $l-1$. The provided equation imply that the output for each node within a layer is based on the outputs from the previous layer. This process is repeated for every layer, showing a method where information moves forward through the network.

3.3.2.2 *Loss function*

Loss function is a method to evaluate the performance of machine learning models, particularly in the context of training neural networks [22]. It quantifies the error or discrepancy between the predicted values ($\hat{y}$) and the actual target values (y) in a dataset. There are generally two main categories of loss functions:

1. Loss Functions for Regression Problems: These functions are typically employed when making predictions for continuous values. One of the most common loss functions is:

   Mean Squared Error (MSE): It calculates the average of the squared differences between the predicted values ($\hat{y}$) and the actual values (y) for each observation m observations. Squaring the differences serves to amplify the impact of larger deviations, penalizing substantial errors more severely. The MSE eqeuation is [23]:

   $$MSE = L(\hat{y}, y) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 \tag{3.6}$$

2. Loss Functions for Classification Problems: These functions are typically employed to quantify the discrepancy between the actual class labels and the predicted class probabilities, providing a way to assess the classification accuracy of the model. One of the most common loss functions for Classification Problems is:

   Cross Entropy Loss: This loss function was designed for binary classification initially, but it can also be extended to accommodate scenarios involving more than two classes. The equation for Binary Cross-Entropy is expressed as follows [24]:

   $$L(\hat{y}, y) = -y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \tag{3.7}$$

3.3.2.3 *Optimization Algorithms*

Optimization algorithms play a crucial role in improving the performance of neural networks by fine-tuning their parameters, which include the weights (w) and biases (b). Optimization algorithms play a crucial role in improving how neural networks operate. Within the context of neural networks, two primary categories of optimization algorithms are first-order and second-order.

- First-order optimization algorithms, work by computing the gradient of the loss function concerning the network's parameters, such as weights w and biases b. These algorithms iteratively adjust these parameters based on the gradient's direction to bring about improvements in network performance. While first-order methods are popular due to their efficiency and scalability, they can sometimes encounter challenges like getting stuck in local optima.
- In contrast, second-order optimization algorithms consider not only the gradient but also the second-order derivatives of the objective function with respect

to the weights and biases. These algorithms offer the potential for faster convergence to optimal solutions, but they tend to be computationally expensive and are often impractical for deep neural networks. Therefore, in the context of neural networks, first-order optimization techniques like stochastic gradient descent and its variants are commonly favored for their ability to strike a balance between effectiveness and computational feasibility while fine-tuning the network's weight and bias parameters.

Gradient descent is one of the most popular first-order optimization algorithms for neural networks. Its primary objective is to minimize the error between the estimated and actual output values by iteratively adjusting the learning parameters, w (weights), and b (bias). It gradually attempts to fine-tune these parameters in order to reduce the cost function. The aim is to find the global minimum through this iteratively fine-tuning process. The equations for Gradient Descent can be stated as follows:

$$w = w - a\frac{\partial J(w,b)}{\partial w} \qquad b = b - a\frac{\partial J(w,b)}{\partial b} \tag{3.8a}$$

,where a denotes the learning rate, and $J(w,b)$ represents the cost function. Typically, the loss function for a single example is denoted as L, while the cost function, which aggregates over all examples, is typically symbolized as J. These functions apply to every weight across all layers and nodes within the neural network. Obtaining the necessary derivatives for these weights is accomplished through the backpropagation algorithm which will be explained in the next subsection.

Examining Fig. 3.4 alongside with Eq. 3.8 , it becomes apparent that when the slope $\frac{\partial J(w,b)}{\partial w}$ is positive, the weight w decreases, resulting in a reduction of the cost function. Through a series of iterative steps, this leads to the eventual discovery of the global minimum. Conversely, the same process unfolds when the slope is negative, effectively minimizing the cost function. It's important to note that the cost function relies on the values of both w and b.

The learning parameter's a role in this method is to control the speed at which the parameters, specifically w and b undergo adjustments. A small learning rate causes the weights to change very slowly during each iteration, which can significantly prolong the learning process since the cost function takes longer to converge. Conversely, if the learning rate is excessively high, it may overshoot the cost function's minimum, leading to imprecise results. Commonly used values for this parameter are 0.1, 0.01, and 0.001. Nevertheless, the optimal value is typically determined through empirical testing on the available dataset.

It's worth noting that there are alternative optimization algorithms such as **RMSprop**, **AdaGrad**, and others, each with its own characteristics and suitability for different scenarios.

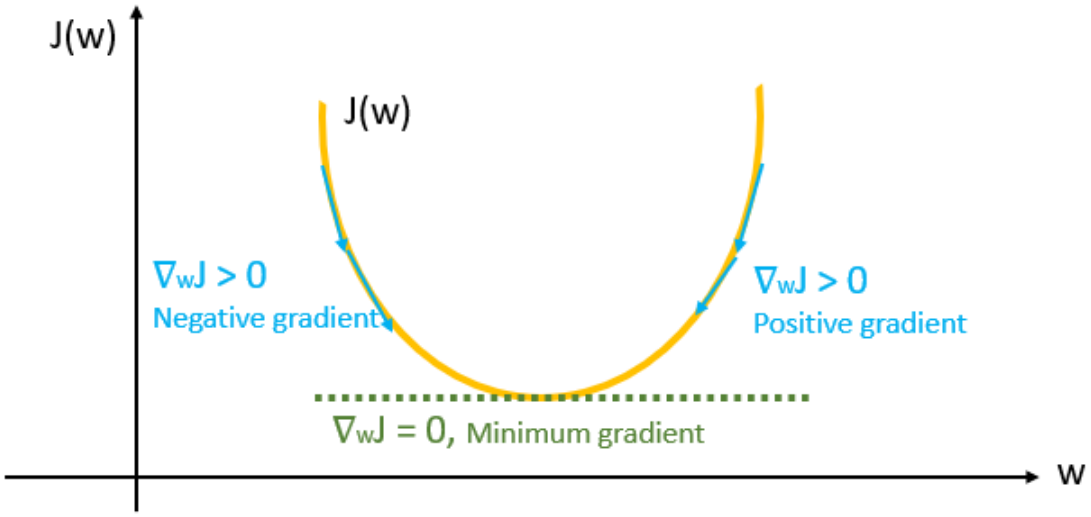

Figure 3.4: A graph showing the cost function in relation to the weights

3.3.2.4 Backpropagation

Backpropagation is a method used in artificial neural networks to update the network's parameters (weights and biases) effectively [25]. It works by computing the gradient of the loss function with respect to each weight by the chain rule, thus making it possible for optimization algorithms to utilize this gradients. The chosen optimization algorithm then adjusts the weights towards a direction that reduces the loss. The process involves two main phases: a forward pass where the input data is passed through the network to generate an output, and a backward pass where the gradient of the loss is propagated back through the network to update the weights. This process repeats for as many iterations, known as epochs in the context of neural networks, required until the loss is minimized. A mathematical representation of backpropagation using the chain rule to compute the gradient of the loss function with respect to the weights is given by:

$$\frac{\partial L(w,b)}{\partial w} = \frac{\partial L(w,b)}{\partial y} \cdot \frac{\partial y}{\partial w} \tag{3.9}$$

, where:

- L, represents the loss function
- w, denotes the weights of the network.
- $y = f(z)$, is the predicted output of the network where f is the activation function.
- $\partial L(w,b)/\partial y$, is the gradient of the loss function with respect to the network outut
- $\partial y/\partial w$, is the gradient of the output with respect to the weights, illustrating how changes in weights affect the network's output.

This algorithm, utilizing the chain rule, initially requires calculating the gradients for every neuron across all layers, systematically working backwards from the output

layer to the input layer. This algorithm, utilizing the chain rule, initially requires calculating the gradients for every neuron across all layers, systematically working backwards from the output layer to the input layer.

The earlier sections introduced the basic training process for a multilayer perceptron (MLP), covering essential steps like forward and backward propagation and weight adjustment. However, training a neural network effectively involves more than these steps. There are additional techniques that help improve a model's performance and its ability to generalize to new, unseen data. Among these techniques are early stopping, which stops training before the model overfits [26], data augmentation, which increases the variety of training data through modifications [27], and L1 and L2 regularization [28], which discourage large weights to reduce overfitting.

While the current section is focused on MLPs, it's important to note that there is also another type of feedforward network, Convolutional Neural Networks (CNNs), particularly useful for processing data with spatial relationships, like images. The details of CNNs and their applications will be explored in the next section.

3.3.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a powerful type of neural network designed to handle data arranged in a grid, such as images. Their architecture is specifically built in order to recognize and learn patterns in data that has spatial connections. Initial layers capture basic patterns like edges and textures, while deeper layers utilize these inputs to identify more complex patterns. Three are the core components of the CNNs, namely **convolutional layers**, **pooling layers**, and **fully connected layers** (FC) which will be analytically explained below. Each layer is contributing to the network's ability to abstract and classify input data effectively.

3.3.3.1 Convolutional layers

In CNNs, input image is processed through convolutional neural layers using filters, or kernels, which is essentially set of weights responsible to scan the entire image with convolution. CNNs filter preserve the image's spacial structure by moving across its width, height and depth. More specifically, consider an input image with dimensions $N_w x N_h x N_d$ representing the (width, height, depth/channels). If the image is RGB then $N_d = 3$. Considering a 2D filter with dimensions fxf that moves over the image to look for the specific features for each channel and add the results. The size of the filter determines the size of the receptive field, or the small area of the image that the filter examines. The filter is applied to the whole image in a window-sliding manner, where the filter values performs dot product with the pixel values of the image and the output is placed in an output array called feature map or activation map. Also it should be noted that the filter itself does not change during training; only the weights within the filter do. This is called Parameter sharing and is a key idea in CNNs. The number of activation maps (output arrays) corresponds to the number of filters employed during the convolution process. Each filter generates one activation map as its output. However, the dimensions of these output activation maps are not only determined by the input image and the filter size. They are significantly

influenced by two additional parameters: **stride** and **padding**. These parameters play a crucial role in defining the spatial size of the output feature map.

- **Stride**: Stride determines the step size the filter takes when moving across the input image. A stride of one (1) indicates that the filter moves one pixel at a time, producing a feature map that nearly matches the spatial dimensions of the input image. On the other hand, a wider stride, such as two (2) or more, increases the space between the positions where the filter is applied, thus reducing the spatial size of the output feature map. In Fig. 3.5 a 3x3 filter moving over a 5x5 input array with a stride of one can be depicted.

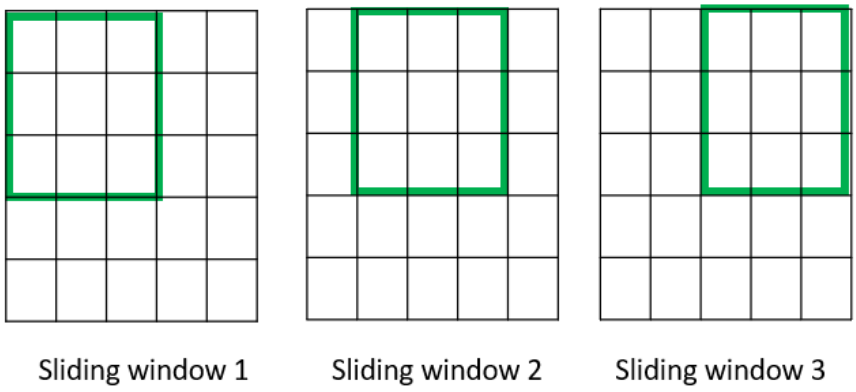

Figure 3.5: A 3x3 filter moving over a 5x5 input array with a stride of one

- **Padding**: Padding is a method used to maintain the input image's spatial dimension after convolution. One of the most common techniques called (zero-padding), is to add zeros around the borders of the input image (Fig. 3.6 to ensure that the convolutional filter can fully cover the edge regions, allowing the output feature map to maintain the original input size. This is especially useful for maintaining the resolution of deeper layers in the network.

The dimensions of a feature map can be found from the following equation:

$$\left[\frac{n_h - f + 2p}{s} + 1\right] x \left[\frac{n_w - f + 2p}{s} + 1\right] \tag{3.10}$$

, where: p and s represent the padding and the stride correspondingly. While Padding and stride can be different for width and height, in Eq. (3.10) they are kept same for simplicity.

3.3.3.2 Pooling Layers

Convolutional layers are followed by **Pooling layers**, which decrease the feature maps' spatial dimensions and, consequently, the network's computation and parameter count. By giving the feature maps a more abstracted form, this helps prevent overfitting. There are two popular kinds of pooling layers:

1. **Max Pooling**: In max Pooling the maximum value from a group of pixels in a window (usually 2x2 or 3x3 filter) is selected and passed to the next layer.

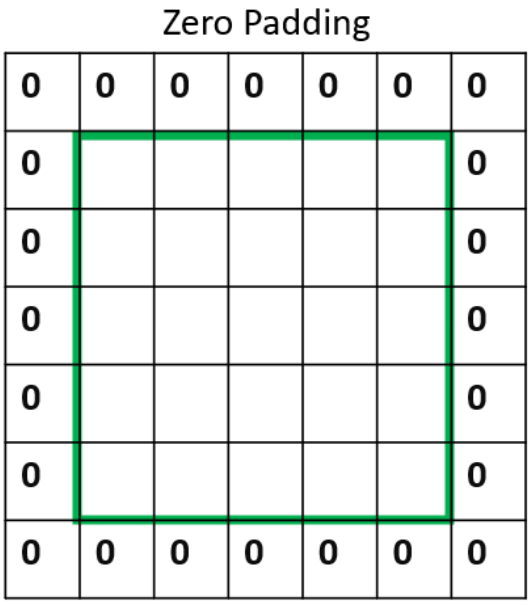

Figure 3.6: Apply zero padding to a 5x5 input to produce a 7x7 output array

This is applied to the whole image in a window-sliding manner and only the dominant pixels are selected and places in an output array. The most notable characteristics found by the convolutional layers are preserved while the feature map's size is decreased.

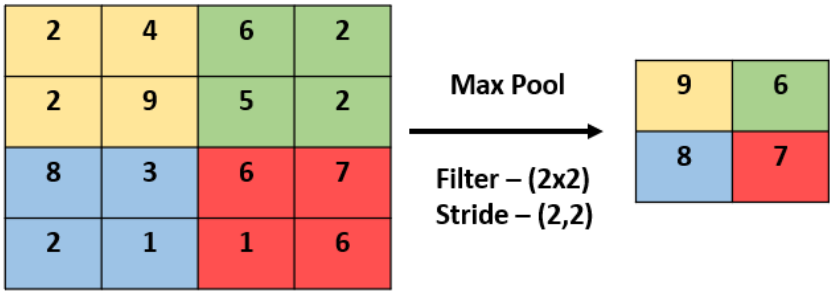

Figure 3.7: Max Pooling applied in a 4x4 image using a 2x2 filter

2. **Average Pooling**: Average Pooling calculates the average of the pixels in the window to provide a more smooth downsampled feature map

Max pooling does not use padding. This means the output dimensions are determined by the size of the pooling window (filter), the stride, and the dimensions of the input feature map. The output dimensions after using pooling layers are determined with the following formula:

$$\left[\frac{n_h - f}{s} + 1\right] x \left[\frac{n_w - f}{s} + 1\right] \tag{3.11}$$

Pooling layers enable the network to concentrate on the most significant features,

making it invariant to minor shifts in the position of the input image, thus being less prone to overfitting.

3.3.3.3 Fully Connected Layers

In a fully connected layer, also known as dense layer, every neuron is connected to every neuron in the previous layer, and each connection has its own weight. In the case of images the pixels are flattened into neurons, with each neuron representing a single pixel value from the input image. This structure allows the layer to learn a high-level representation of the input data by combining features. Fully connected layers in neural networks are typically the ending layers of a neural network helping to integrate learned features from previous layers and make final predictions or classifications. While the are capable of capturing complex patterns, due to their fully connected nature they are computationally expensive and prone to overfitting, especially when dealing with large input sizes such as images. For this reason, employing convolutional layers in the initial layers of a network is preferred because of their utilization of shared-weight filters across the input, which minimizes the number of parameters and decreases the likelihood of overfitting. Meanwhile, fully connected layers are reserved for the final layers to integrate and process the learned features.

3.4 Object Detection Technologies

3.4.1 CNNs towards object detection

In traditional machine learning approaches for object detection, a significant emphasis is placed on the process of feature engineering, where specific characteristics or attributes of the objects are manually identified and used for detection. However, deep learning methods diverge from this approach. They learn and figure out these important features by themselves by analyzing large amounts of data. Because of this self-learning ability, deep learning often outperforms traditional machine learning, especially when it has a lot of data to learn from. This makes deep learning a more efficient and effective choice for object detection tasks. Before the advent of YOLO (You Only Look Once) models, Convolutional Neural Networks (CNNs) were the primary framework used for object detection in computer vision. This period saw the development of several key CNN-based models, each contributing to the evolution of object detection techniques:

1. **R-CNN (Region-based Convolutional Neural Networks)**: Developed by Girshick et al. in 2014 [29], the R-CNN model was a pioneering approach in object detection. It utilized the selective search algorithm [30] to identify potential object locations in an image, effectively proposing regions where objects might exist. This method combined similar pixels and textures within various rectangular frames. The algorithm suggested around 2000 such frame positions, which were then processed by a pre-trained convolutional neural network (CNN). The output feature maps from the CNN were fed into SVMs (Support Vector Machines) [31] for object classification. This approach was a significant step forward in identifying objects within images, as it combined region proposals with deep learning. In R-CNN, bounding box regression is

used to refine the position and size of each detected object's bounding box, improving localization accuracy. For each object class, a distinct regression model is trained to adjust the initial bounding box to better align with the actual object contours. Following classification and bounding box regression, R-CNN applies Non-Maximum Suppression (NMS). NMS filters out redundant and overlapping bounding boxes, retaining only those with the highest confidence, ensuring precise and distinct object detection. Fig. 3.8

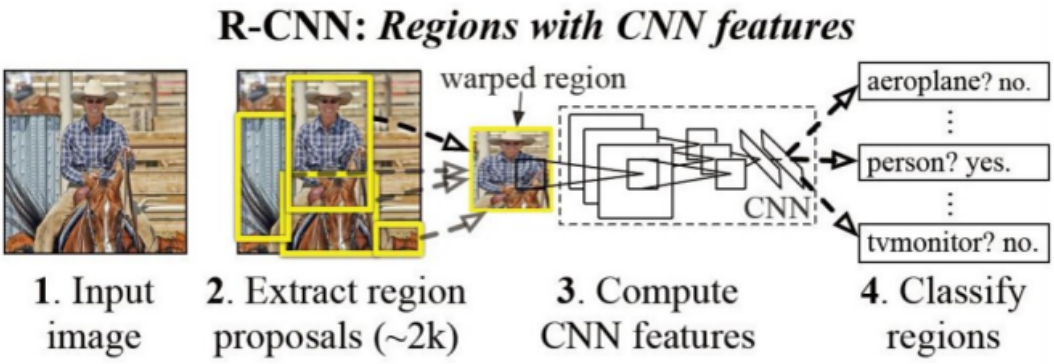

Figure 3.8: RCNN architecture [29]

2. **Fast R-CNN (2015):** Fast R-CNN, introduced by R. Girshick [29], was an improvement over the original R-CNN. This model streamlined the process by passing the original image only once through a pre-trained CNN. Instead of processing 2000 proposed regions individually, Fast R-CNN used the output feature map from the CNN for region proposal calculation. It introduced the ROI (Region of Interest) pooling layer to ensure a standard and predefined output size. These outputs were then passed to fully connected layers. The final stage involved two output vectors: a softmax classifier to predict the class of the observed object and a linear regressor to adjust the bounding box coordinates. Fast R-CNN improved efficiency and speed by optimizing the process of region proposal and feature extraction.

3. **Faster R-CNN (2015):** Faster R-CNN [32], an advancement over Fast R-CNN, brought further improvements. The major innovation in Faster R-CNN was replacing the selective search algorithm with a Region Proposal Network (RPN) (Fig. 3.10). RPN is a convolutional network that significantly sped up the computing time for region proposals. This change allowed Faster R-CNN to generate region proposals more efficiently, making the model faster and more accurate in detecting objects. The integration of RPN into the Faster R-CNN framework streamlined the object detection pipeline, enhancing both speed and performance.

Building on the foundations laid by R-CNN, Fast R-CNN, and Faster R-CNN, the YOLO series introduced a groundbreaking shift in object detection techniques:

1. **YOLOv1 (2015)**: "YOLOv1, short for 'You Only Look Once', introduced a groundbreaking method in object detection by Joseph Redmon and his team [33]. Unlike previous models that analyzed different parts of an image step-by-step for detecting objects, YOLOv1 revolutionized this process. It scans the

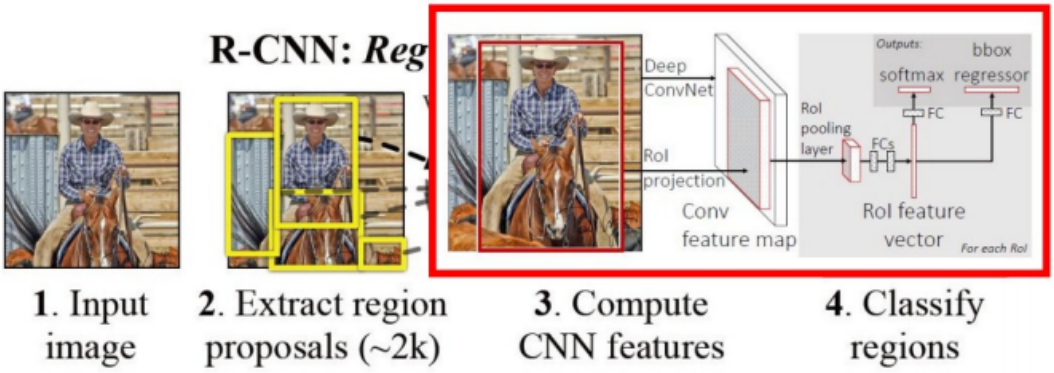

Figure 3.9: The Fast R-CNN architecture processes an input image and various regions of interest (RoIs) through a fully convolutional network. Each RoI is transformed into a fixed-size feature map, which is then converted into a feature vector by fully connected layers (FCs). This network produces two outputs for each RoI: softmax probabilities for class predictions and per-class bounding-box regression offsets. Fast R-CNN is trained with a multi-task loss, optimizing it end-to-end for both classification and precise bounding box predictions. [29]

whole image in just one attempt. This single-step process significantly sped up how quickly objects could be detected, making it possible to do this in real-time, a huge leap forward from previous methods.

The way YOLOv1 works is by dividing the image into a grid. Each section of the grid is responsible for identifying objects and predicting their categories (Fig. 3.11). By doing everything at once - locating objects and figuring out what they are - YOLOv1 made the whole detection process simpler and faster. However, it wasn't perfect. YOLOv1 found it challenging to recognize smaller objects and sometimes got confused when objects overlapped.

2. **YOLOv2 (2016)**: Developed by Joseph Redmon and his team [34], YOLOv2, or YOLO9000, built upon the foundation of YOLOv1 with significant enhancements. One of its key features is multi-scale training, which allows the model to adjust its size, giving users the flexibility to choose between faster detection speed and higher accuracy. YOLOv2 also introduced the concept of anchor boxes. These are predefined box shapes used to improve the accuracy of predicting the size and shape of objects. To optimize these anchor boxes, YOLOv2 applied k-means clustering on the bounding boxes within its training set, leading to more precise shape predictions. Furthermore, YOLOv2 greatly expanded its detection capabilities, now able to identify over 9000 different object categories. This vast expansion made YOLOv2 much more versatile in recognizing a wide variety of objects. These combined improvements significantly boosted YOLOv2's performance, making it a more adaptable and effective tool for object detection tasks.

3. **YOLOv3 (2018)**: YOLOv3 which introduced by Joseph et. al [35], further enhanced the architecture by predicting at three different scales, making it more adept at detecting small objects. It used logistic regression for objectness scores and trained binary classifiers for multiple labels, showcasing improved

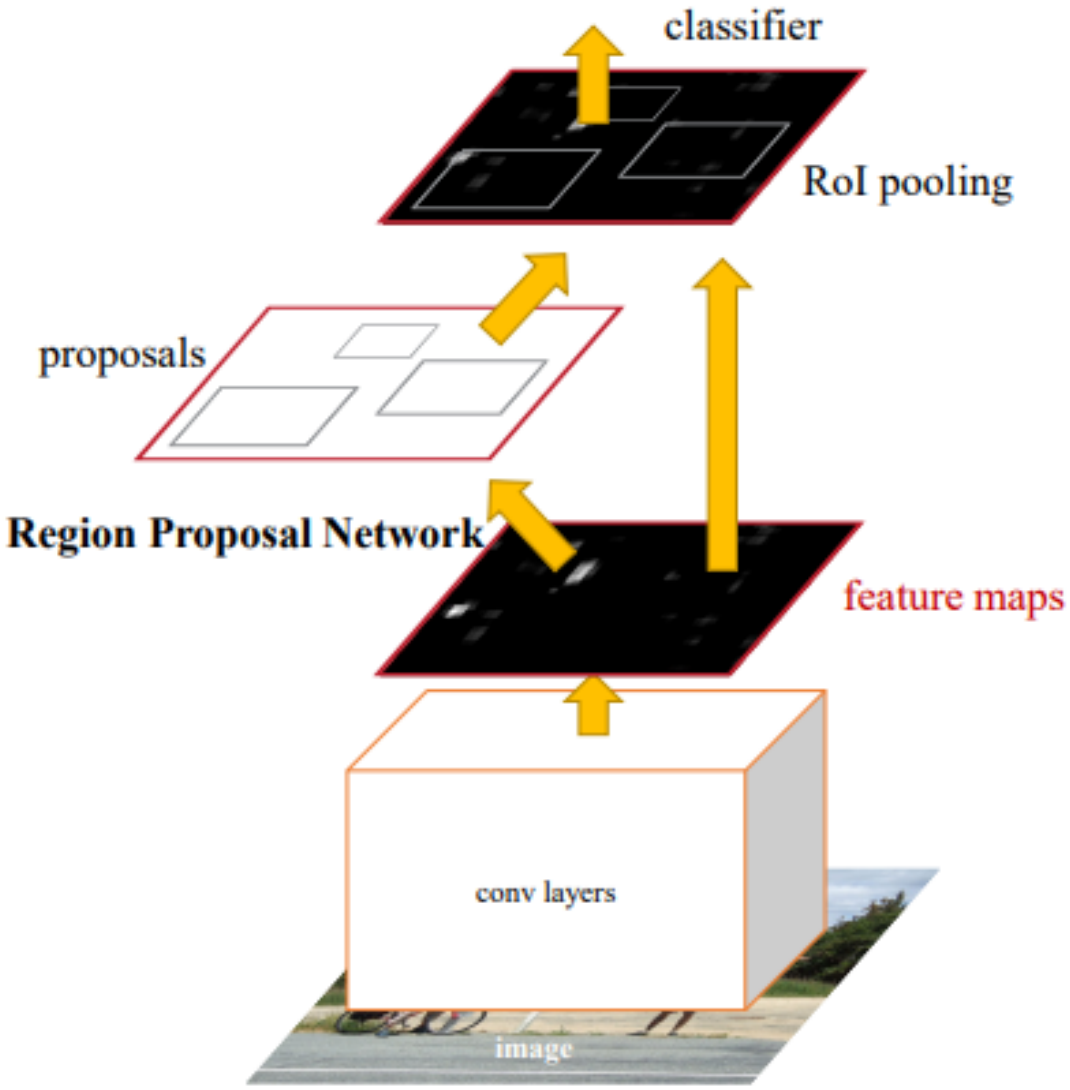

Figure 3.10: Faster-CNN architecture [32]

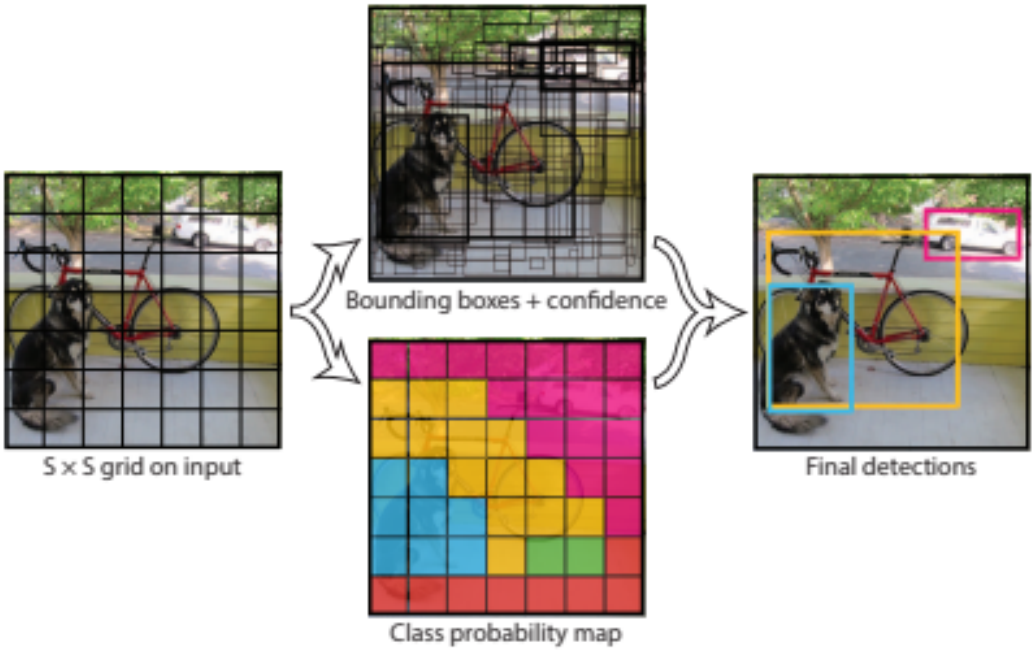

Figure 3.11: Yolov1, Unified Detection [32]

performance on the MS COCO dataset.

4. **YOLOv4 (2020)**: After Joseph Redmon stopped developing YOLO, Alexey and his team took over, introducing YOLOv4 [36]. This version focused on achieving optimal speed and accuracy, using methods like bag-of-freebies and bag-of-specials to enhance performance.

5. **YOLOv5 (2020)**: YOLOv5, created by Glenn Jocher and his team at Ultralytics [37] , emerged as a significant update to the YOLO series. Unlike YOLOv4, YOLOv5 was developed using PyTorch, a different technology than the previous DarkNet framework. This change led to some controversy, especially because it was released without an accompanying research paper. Despite this, YOLOv5 stands out for its impressive performance, particularly on widely-recognized datasets like MS COCO. It strikes an excellent balance between speed and accuracy, making it well-suited for real-time object detection tasks. In terms of its design, YOLOv5 shows marked improvements in how it predicts the size and position of objects (bounding box regression), leading to more accurate detection results.

6. **YOLOv6 (2020)**: Released by Meituan [38], YOLOv6 is a significant evolution in the YOLO object detection series, introducing key architectural changes like the decoupled head design for enhanced hardware efficiency. This design separates classification and localization processes, leading to more focused and accurate object detection. A major innovation in YOLOv6 is its anchor-free detection method, which speeds up the process by about 51% compared to traditional anchor-based methods. This speed increase is due to the model's reduced reliance on predefined anchor points, allowing it to detect objects more quickly and with fewer constraints. Evaluated on the COCO dataset,

YOLOv6 demonstrates impressive improvements in accuracy and speed over its predecessors, such as YOLOv5. This performance enhancement is vital for real-time applications where rapid and reliable object detection is crucial. YOLOv6 is especially suitable for industrial applications, where its efficient hardware usage and quick detection capabilities are essential.

7. **YOLOv7 (2022)**: Developed by Wang et. al [39], YOLOv7 focused on trainable optimization methods and modules to improve accuracy. while maintaining the core objective of enhancing real-time detection capabilities. Specificaly, YOLOv7 implements a strategy called 'trainable bag-of-freebies', which focuses on optimizing the network's structure and loss functions. This approach is designed to boost accuracy without sacrificing the speed of detection, addressing one of the key challenges in real-time object detection.

8. **YOLOv8 (2023)**: Ultralytics [39], the developers behind YOLOv5, introduced YOLOv8, which featured semantic segmentation capabilities and anchor-free detection, marking another advancement in the YOLO series.

Throughout its evolution, each YOLO version has strived to balance speed, accuracy, and efficiency, continually pushing the boundaries of real-time object detection. The series has evolved from a novel, unified approach in YOLOv1 to a sophisticated, multi-faceted framework in YOLOv8, demonstrating significant strides in computer vision and deep learning technologies.

3.4.2 ResNet for Object Categorization

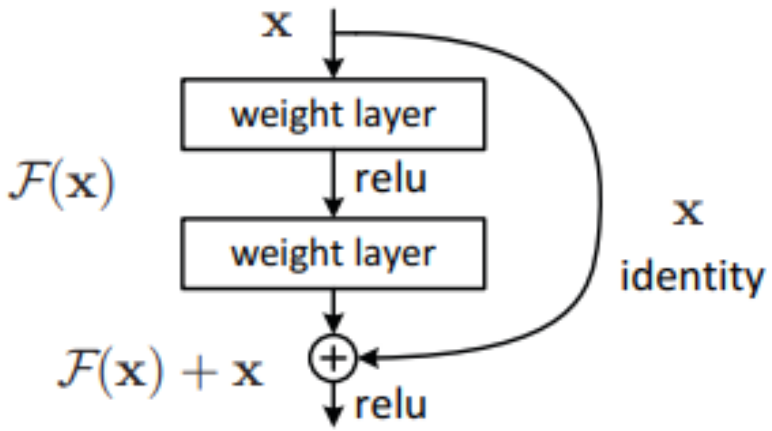

Figure 3.12: Resnet's building block [40]

Residual blocks are a fundamental component of Residual Networks (ResNets) [40], a class of DCNNs and were introduced to address the problem of training very deep networks. Traditional DCNNs suffer from the vanishing gradient problem, which makes them hard to train as they get deeper. The issue is that, during backpropagation, gradients often get smaller and smaller, causing the weights in the early layers to update very slowly. To overcome this problem ResNet introduced the concept of **residual learning**. In a ResNet's residual block, each layer learns not the full output, but the small difference (residual) between the input and the the block's output. The

key innovation in residual blocks is the introduction of **skip connections** (also called shortcut connections). These skip connections bypass layers, adding a layer's input directly to its output. For input x and desired output H(x) layers learn the difference F(x) = H(x) - x. The final output is F(x) + x (Fig. 3.12), meaning learned changes with the original input. With skip connections, the network can easily learn the identity function, which ensures that adding more layers does not degrade performance making it easier to train deeper networks since the gradient can flow directly through the skip connections, mitigating the vanishing gradient problem. A typical residual block in a ResNet has two or three convolutional layers. The input to the block is added to the output of these layers before applying the activation function. This addition operation is the skip connection. In more complex variations, the skip connection might involve additional operations (like a convolution) to match the dimensions of the input and output.

4 Methodology

4.1 System Design and Architecture

In this chapter, the focus is on the system design of the tool aimed at evaluating the dementia-friendliness of indoor environments. This section is organized to give a clear overview of how the tool operates, breaking down its architecture and the function of each component. The chapter begins by laying out the big picture, explaining how each part of the tool contributes to its effectiveness. Then, it takes a deep dive into each component, explaining their purpose and how the function together, and why they matter in making indoor spaces better for individuals with dementia, especially focusing on flooring and rugs. Lastly, the chapter talks about containerizing these components, which is important for making the tool more scalable, maintainable and easier to extend.

4.1.1 Overall System Framework

The system architecture of the web platform is designed to provide a comprehensive solution for assessing indoor spaces in terms of their friendliness to individuals with dementia. As depicted in Fig. 4.2, the architecture comprises four key components: the Client App/UI (built with ReactJS), the Image Detection Server (implemented in Flask-Python), the Authentication Server (developed using Node.js), and the Image File Storage Server (also built with Node.js). Before delving into the detailed descriptions of each component and their functionalities, Figure 4.1 provides a high-level overview of the system's architecture. This diagram illustrates the interconnection of the four main components—the Client App/UI, Image Detection Server, Authentication Server, and Image File Storage Server—highlighting the flow of HTTP requests, RESTful server interactions, and the database's role in the ecosystem.

The **Client App/UI** serves as the primary interface for users, supporting functionalities such as user authentication, photo uploading, object annotation, and image

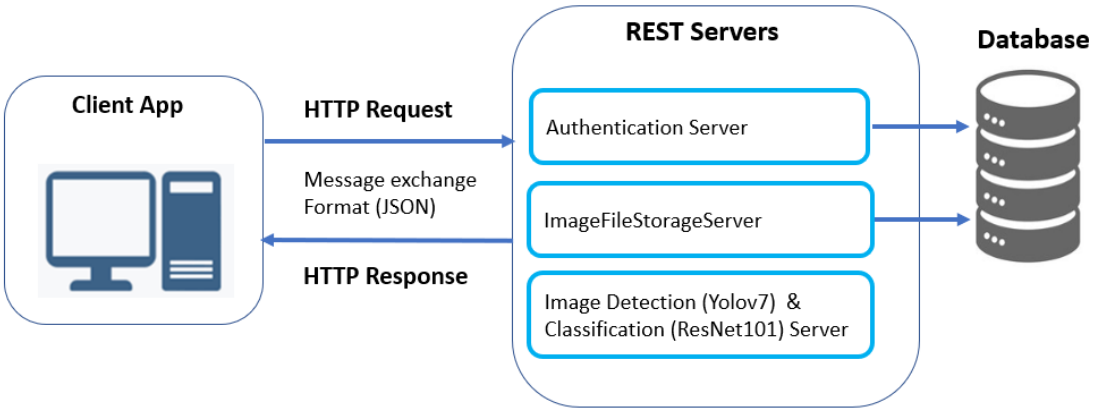

Figure 4.1: Overview of the High-Level System Architecture

labeling. Through this interface, users can upload photos of indoor spaces from various environments, including houses, care homes, and other buildings. Furthermore, the app facilitates object annotation, allowing users to draw bounding boxes around objects and categorize them into specific classes (carpets, mirrors and other interior objects). Users can assign labels to uploaded images to indicate the location where the photo was taken (e.g House, Care Home), aiding in categorizing their images into folders based on location types. Furthermore, the Client App/UI presents aggregated statistics for each location type per user. These statistics provide insights into the level of dementia-friendliness of the images within a specific location type, such as houses or care homes, by analyzing the distribution and classification of objects and their attributes in the uploaded photos. These statistics count the number of objects identified in each class within the uploaded photos and assess how many of these objects are deemed dementia-friendly. This process results in providing a percentage of dementia-friendliness for each object class within each location type, offering a detailed assessment of the environment's suitability for individuals with dementia.

The **Image Detection/Classification Server** plays a crucial role in analyzing images of indoor spaces uploaded by users, using YOLOv7 technology for object detection to evaluate whether these objects meet dementia-friendly standards. Within this framework, objects that are suitable for dementia-friendly environments are labeled as positive samples, while those deemed unsuitable are classified as negative examples. Initially, this server detects and evaluates objects to determine their appropriateness for environments designed to be dementia-friendly.

The system can be further refined by user input. If the automatic detection process fails to identify certain objects, users can manually highlight these objects on the website. This involves users drawing boxes around the objects and assigning them to specific object classes for which a pretrained network is available, aimed at determining the object's suitability for a dementia-friendly environment.

For each object class identified by users, a pretrained classification network that uses Resnet-101 is utilized to reevaluate these manually highlighted objects. Currently, the system's pretrained network is exclusively focused on carpets. Therefore, as the server's capabilities expand to include additional objects for dementia-friendly

assessment, these new categories will be made available on the website for user selection.

The **Authentication Server** is responsible for managing user authentication and authorization processes securely. It stores session information in the server and user credentials in a MongoDB database.

Lastly, the **Image File Storage Server** serves as a centralized repository for storing uploaded images and associated metadata. Utilizing MongoDB, this server maintains associations between images and their metadata, including location labels, user information, and other relevant data. Additionally, the server calculates aggregated statistics on uploaded images, providing valuable insights into the distribution of objects and their metadata across different locations. The server carefully counts the objects in each category from the uploaded images and determines whether they are suitable for dementia-friendly environments. Using this information, it calculates a score for how dementia-friendly each category of object is across various locations. This gives a clear overview of how well different environments meet the needs of individuals with dementia. This approach helps us better understand how supportive and accessible these spaces are for people with dementia.

In summary, the system architecture outlined above establishes a framework for assessing indoor spaces' friendliness to individuals with dementia. Through a combination of user-friendly interfaces, object detection and authentication mechanisms, and data storage solutions, the platform empowers users to contribute to dementia research and caregiving efforts effectively. The following subsection delves into a detailed explanation of each component to provide a thorough understanding.

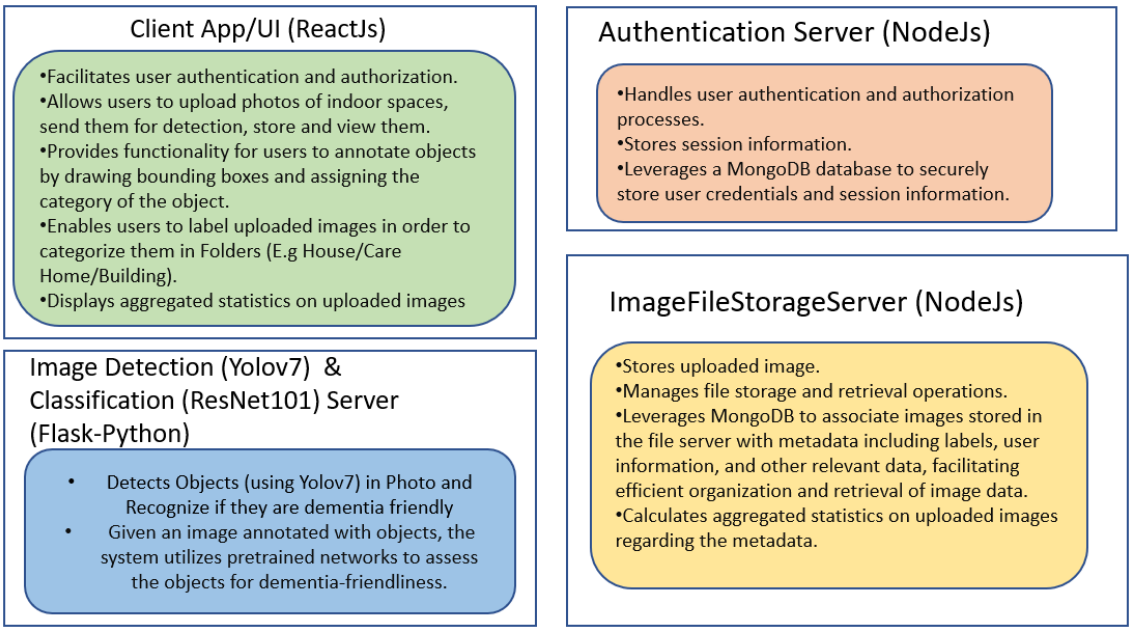

Figure 4.2: System Component Architecture

4.2 Implementation Details

4.2.1 Database Schema Design

The evaluation system's ability to effectively assess the dementia-friendliness of indoor environments depends on a designed database schema, utilizing MongoDB for its scalable and data management capabilities. This subsection outlines the schema's structure, comprising two main collections: `Users` and `UserImages`. Before delving into thorough details of each component of the tool developed, the important role of the database schema is highlighted, as illustrated in Fig. 4.3.

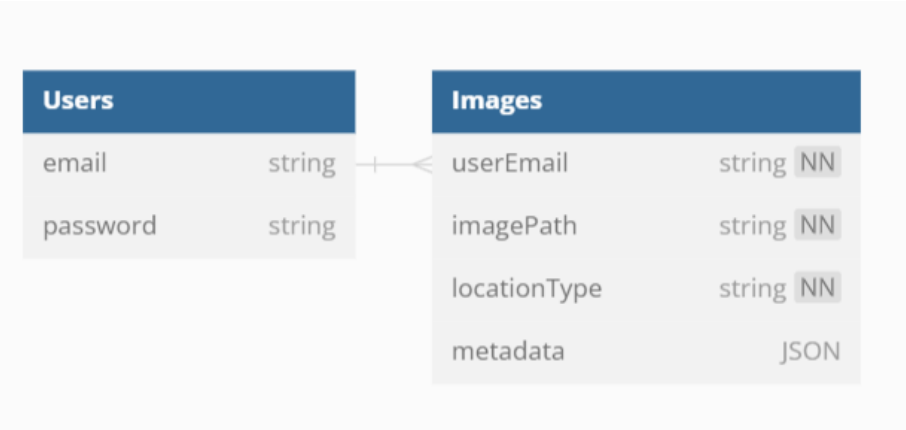

Figure 4.3: Database Schema Overview

Users Collection: The `Users` collection is crucial for managing user-specific data, including authentication credentials and session information. Each user record is uniquely identified by an email address, ensuring secure and personalized interactions within the system.

UserImages Collection: The `UserImages` collection holds detailed metadata for each image uploaded by users. Key fields in this collection include:

- `LocationType` – Specifies the environment from which the image was taken (e.g., House, Care Home), helping to sort images for further analysis.
- `Path` – Indicates the file storage path of the image, enabling efficient retrieval.
- `UserEmail` – Links the image to the uploading user, facilitating user-centric data management.
- `Metadata` – A JSON object that includes keys for all detected objects within the image. Each key has subkeys for the counts of positive (dementia-friendly) and negative samples of each object. This structure is essential to calculate statistics and assessing the dementia-friendliness of environments.

4.2.2 Client App (React)

The Client Application serves as the user interface and is the primary portal through which users engage with the system. Its capabilities are visually represented in

Figure 4.4. While the functional aspects of the Client App were briefly covered in the preceding chapter, this section aims to delve into the Client Application's connectivity with the backend components and will also outline the folder structure of the project's directories.

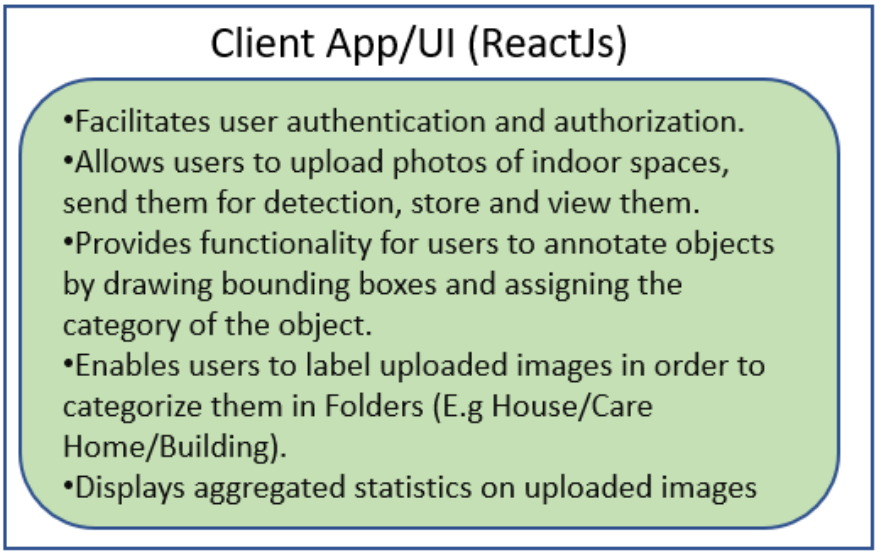

Figure 4.4: Client Component

4.2.2.1 Interactions with Backend Components

This section describes the user's actions within the front-end interface and how these actions relate to and interact with each backend service. For each user action, there is an associated activity diagram that illustrates the process flow.

1. **Interactions with the Authentication Server:** This will describe how the Client App interfaces with the Authentication Server to handle user logins, sign ups and sessions. The interactions include:
   - Users can **sign in**, which verifies their credentials against the MongoDB database (Fig. 4.5).
   - Users can **sign up**, creating a new account that is stored within the MongoDB database (Fig. 4.6).
   - Users can **sign out**, effectively deleting their session data from the server to maintain security (Fig. 4.7).
2. **Interactions with the Image Detection Server:** This item will explain how the Client App sends images to the Image Detection Server and processes the detection results, with two key user actions:
   - Users can **send a picture** to the server, which then performs image detection using YOLOv7 to identify the class of objects present (e.g., carpet, mirror, door, table) and assess if they are dementia-friendly (Fig.4.8).
   - If the YOLOv7 detection model is unable to detect certain objects, users have the capability to manually assign these objects one of the predefined

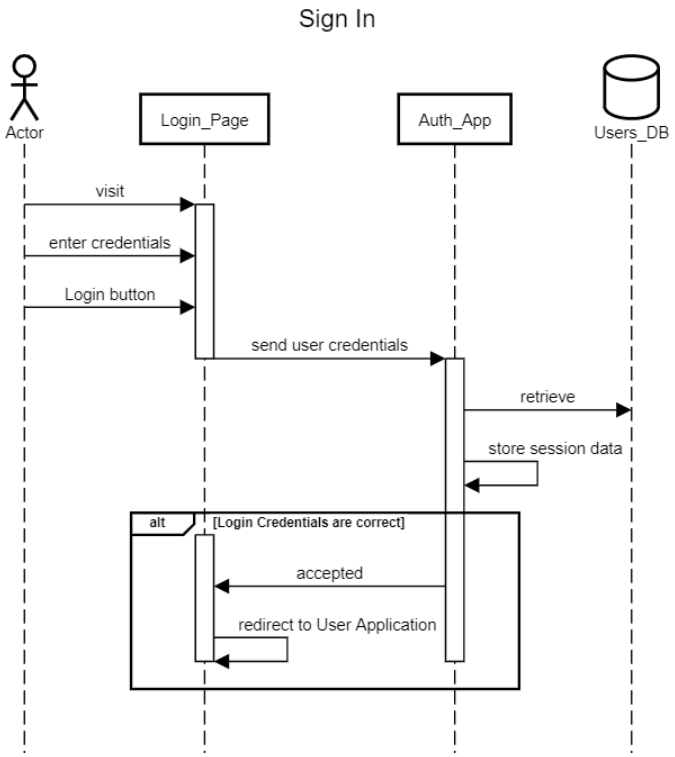

Figure 4.5: Login Sequence Diagram

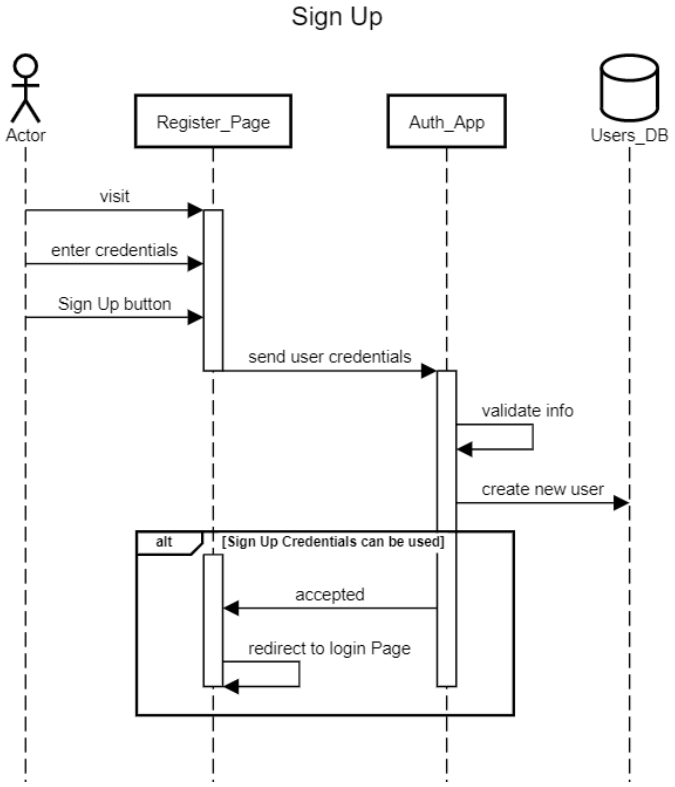

Figure 4.6: Sign up Sequence Diagram

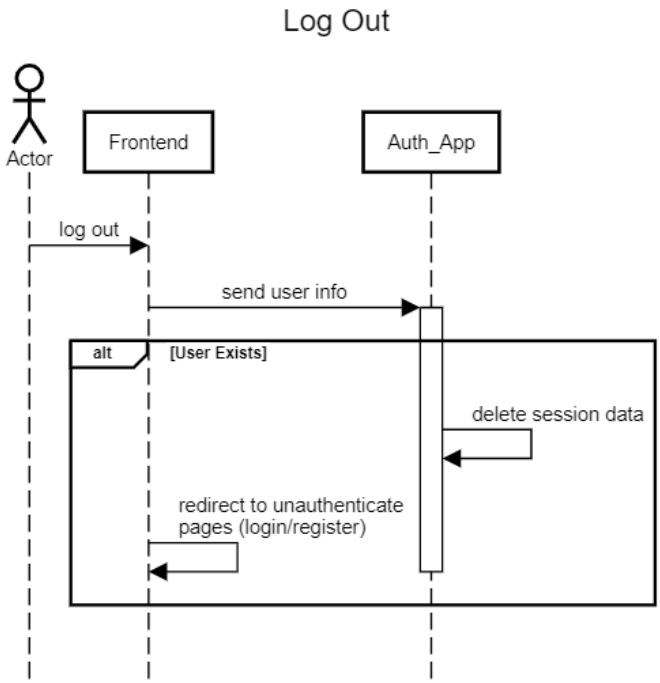

Figure 4.7: Logout Sequence Diagram

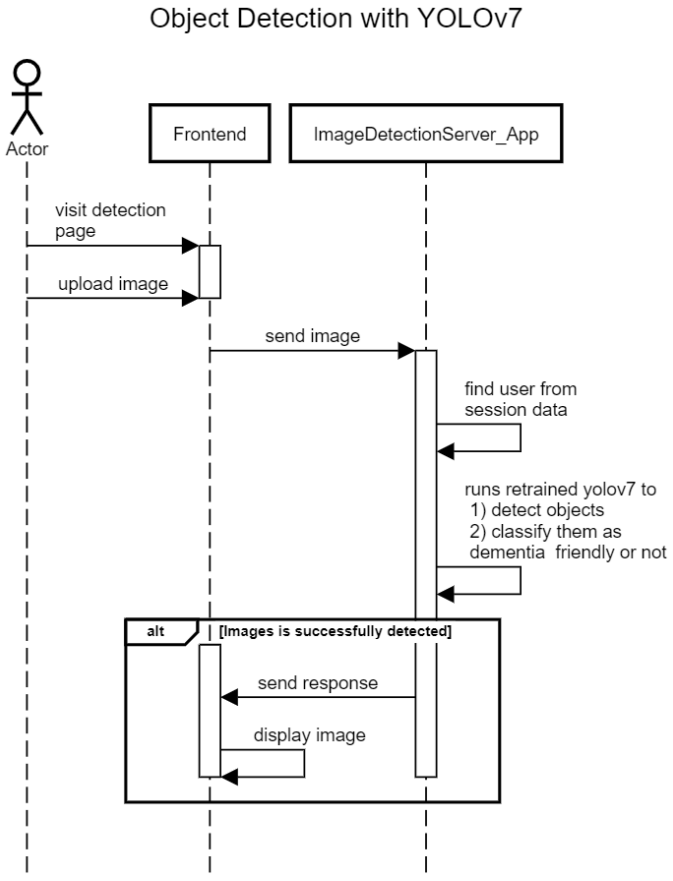

Figure 4.8: Detection with Yolov7 Sequence Diagram

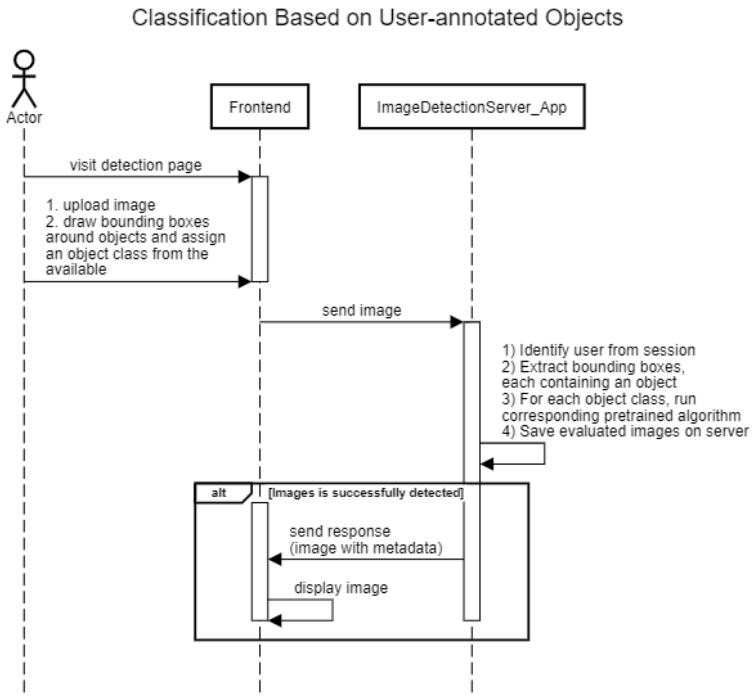

Figure 4.9: Object Classification Sequence Diagram

classes available on the website, assuming they match one of these categories. These predefined classes correspond to pretrained classification networks designed to evaluate whether objects within these categories are appropriate for dementia-friendly environments. Upon this manual classification, the Image Detection/Classification Server bypasses the detection process and instead employs the pretrained networks for the selected object classes to evaluate the objects' suitability for dementia-friendly settings (Fig. 4.9).

3. **Interactions with the Image File Storage Server:** This section details how users, through the Client App, engage with the Image File Storage Server for managing images. Key functionalities provided to users include:

   - **Uploading and storing detected images:** After detection, users can upload images, which are then stored on the Image File Storage Server (Fig. 4.10).
   - **Categorizing photos by location type:** Users have the ability to organize their photos based on the location they represent (e.g., home, care home).
   - **Viewing and deleting uploaded photos:** Users have the capability to navigate through their collection of uploaded images, which are now organized by location type (e.g., home, care home), allowing for easier access and management (Fig. 4.11). They can also remove any photos from the server that they deem unnecessary (Fig. 4.12) .
   - **Accessing statistics:** The platform facilitates users in obtaining comprehensive statistics for their images, segmented by location type. This includes insights into the distribution of various object classes (e.g., mirror, carpet, door) within these images and the proportion of objects within

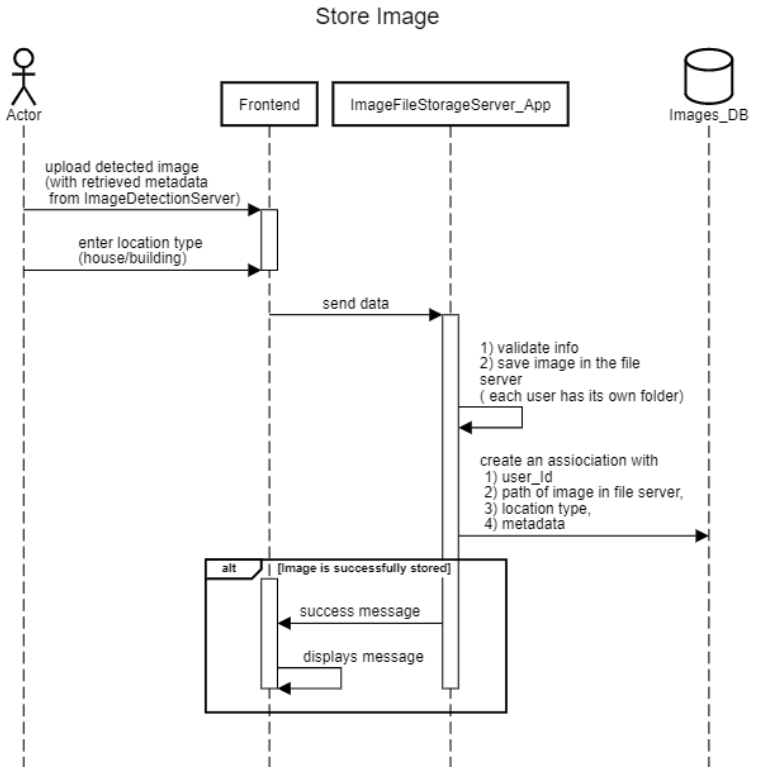

Figure 4.10: Store Photo Sequence Diagram

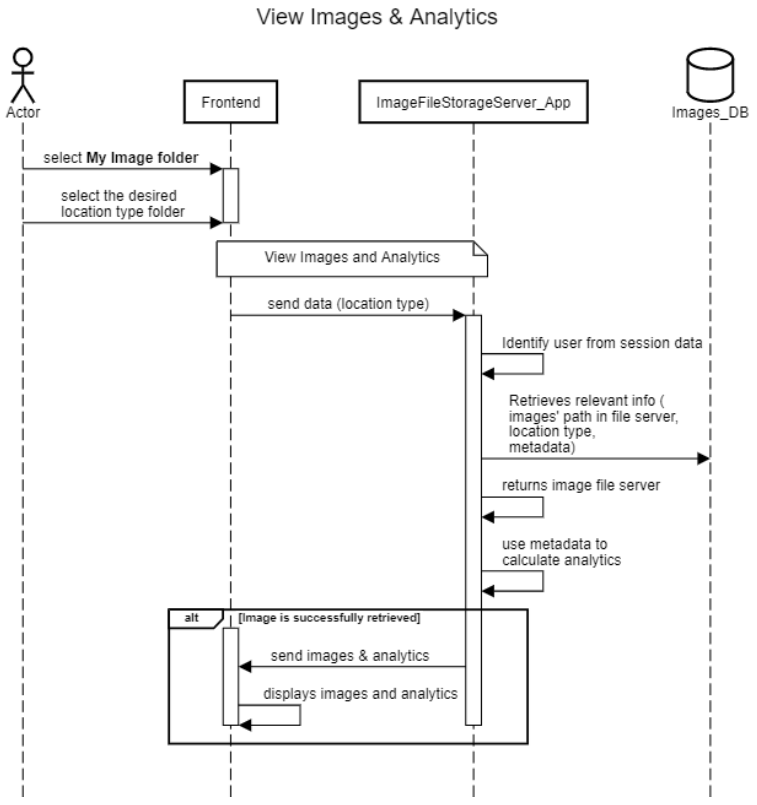

Figure 4.11: View Photo & Analytics Sequence Diagram

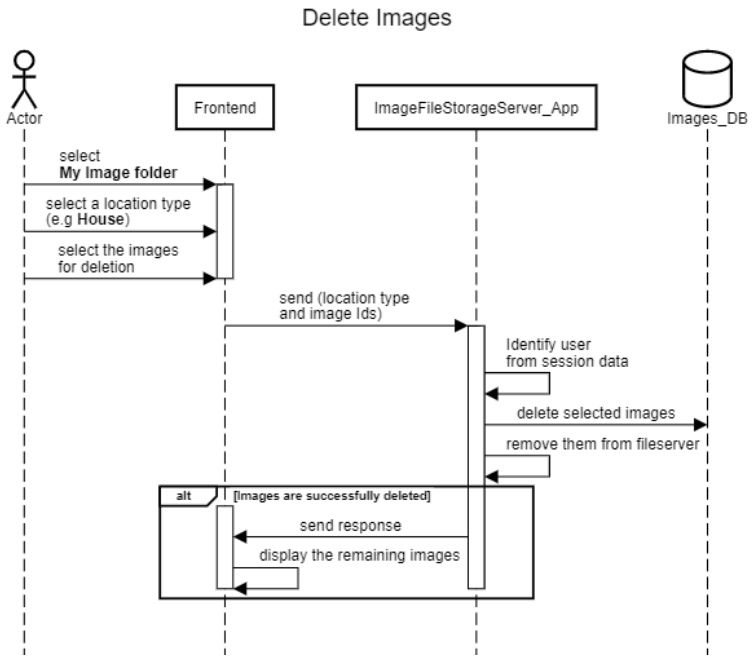

Figure 4.12: Delete Photo Sequence Diagram

each class that are considered dementia-friendly (Fig. 4.11).

It is important to note that a MongoDB database is used to associate image IDs, stored on the File Server, with user information. This association facilitates the retrieval of images based on the user and the location type, along with the metadata returned from the Image Detection Server.

This section outlines the user's actions within the front-end interface and how these actions correspond to and interact with each backend service. This approach offers a narrative-driven exploration of system functionality, which complements the technical diagrams provided later.

4.2.2.2 Folder Structure

The structure of the client application, specifically developed with React, is designed to enhance maintainability, scalability, and clarity in development. The organization of directories is pivotal for separating functionalities and ensuring that components, utilities, and services are easily identifiable and accessible. The following outlines the primary directories within the React application and their designated roles:

- **assets:** This directory contains various static resources such as logos, images, and icons, primarily in SVG or PNG formats. It serves as a central repository for all graphical elements used across the application.
- **components:** Includes reusable React components that can be utilized across different parts of the application. This approach enhances reusability and helps future development by leveraging existing UI elements for new features.
- **context:** State management mechanisms are implemented in this folder, such as authentication contexts. It enables efficient state sharing across components without prop drilling simplifying state management and enhancing com-

ponent reusability.

- **layouts:** Contains layout components such as UserLayout, AuthLayout that contain the header, footer, and sidebar components. These components define the structural layout of different pages within the application.
- **pages:** Contains implementations for each application page, organized by route structure for easy navigation and maintenance.
- **routes:** Manages the configuration of application routes, including the definition of url and their corresponding component renderings.
- **services:** This directory contains functions and utilities for making HTTP requests to backend services. It abstracts the API calls from the components, promoting cleaner code and separation of concerns between the UI logic and data fetching/manipulation.
- **utils:** Comprises reusable utility functions that can be employed across the application, enhancing reusability and reducing code duplication.

It should be noted that tests are integrated directly within the folder of each component, rather than being located in a separate primary directory. This arrangement not only enhances maintainability but also simplifies the process of locating tests.

4.2.3 Authentication App (Backend)

The Authentication App is the secure gateway of the system, handling user sign-ins/ sign-up/ sign-out and maintaining session data. It ensures that user information is safely stored and managed through a MongoDB database. Fig. 4.13 below illustrates the essential functions of the system.

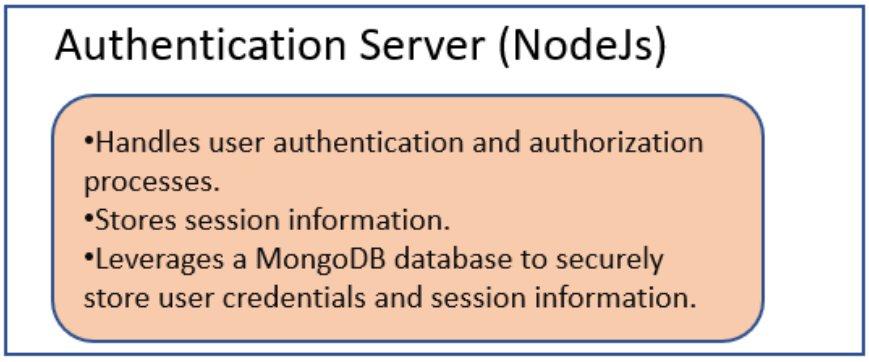

Figure 4.13: Authentication Component

4.2.3.1 Folder Structure

The Authentication Server, built using Node.js, is organized into several key directories, each with a specific purpose, to ensure a clean and maintainable codebase. The following describes the core directories and their function:

- **middlewares:** This directory contains middleware functions necessary for the operation of the authentication server specifically.
- **models:** Hosts the MongoDB database schema definitions. These models structure the user data and session information that the server handles.
- **routes:** Contains the implementation of the server's API endpoints. Each route defines the HTTP methods (GET, POST, etc.) available and the corresponding business logic, facilitating user authentication and authorization processes.
- **test:** Includes automated test suites for the backend of the authentication app, ensuring that the authentication processes are robust and perform optimally under a variety of use cases.
- **utils:** Comprises utility functions that are reused across the application. This centralization of common logic enhances code reusability and maintainability.
- **Common Errors and Middlewares:** Instead of being located within the server's directory, **common errors and middleware functions have been abstracted into a separate NPM library** created for this purpose. This modular approach allows for these components to be maintained separately from the main codebase and reused across multiple projects. The current strategy was inspired by a Udemy instructor Stephen Grider [41].

4.2.4 ImageFileServer App (Backend)

The ImageFileServer App is a critical component of our backend infrastructure, responsible for handling all image-related transactions. Details on the various functions and endpoints provided by this server have been outlined in the previous section. For a visual representation, the relevant functions can be depicted in Figure 4.14. This app employs a directory structure similar to that of the Authentication App, ensuring consistency and maintainability across our backend services.

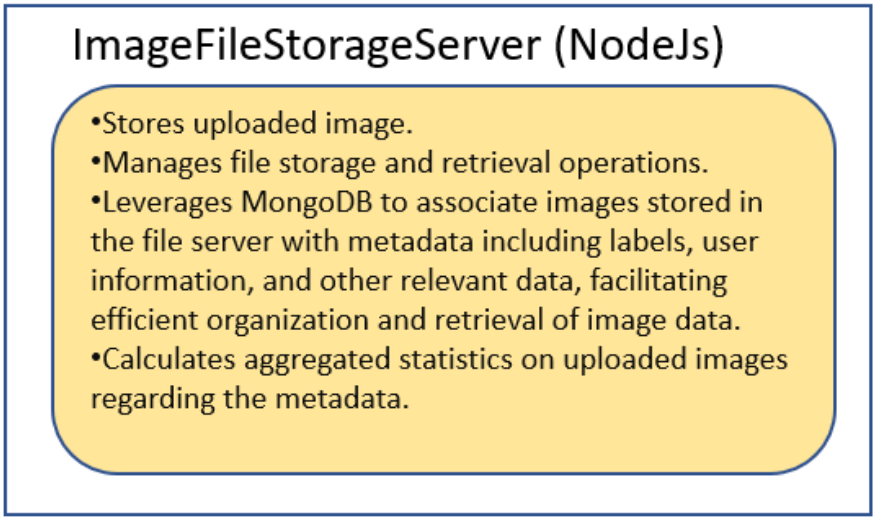

Figure 4.14: Image File Storage Component

4.2.5 Image Detection/Classification App (Backend)

The Image Detection server is a crucial component of the system, integrating deep learning models to evaluate whether objects in photographs are dementia-friendly. The primary functionalities of this server are visualized in Fig. 4.15, which provide a comprehensive outline of its operations. Key functions include:

1. **Object Detection with YOLOv7:** Utilizing the robust YOLOv7 algorithm, retrained with a custom dataset, the server identifies objects in images and evaluates their appropriateness for environments frequented by individuals with dementia, such as care homes and residential settings. This process involves recognizing objects and assessing them to categorize as dementia-friendly or not, based on their suitability in these sensitive environments.
2. **Object Labeling and Classification utilizing Resnet-101:** Users can annotate images, assigning them to one of the available classes predefined in the system. These available classes correspond to categories for which a ResNet-101 network has been retrained, enabling the classification of objects as either suitable or unsuitable for dementia-friendly environments. Upon receiving these annotations, the server employs a specific classification algorithm tailored to each class. This approach ensures that each item is evaluated with the most appropriate algorithm for accurate assessment.

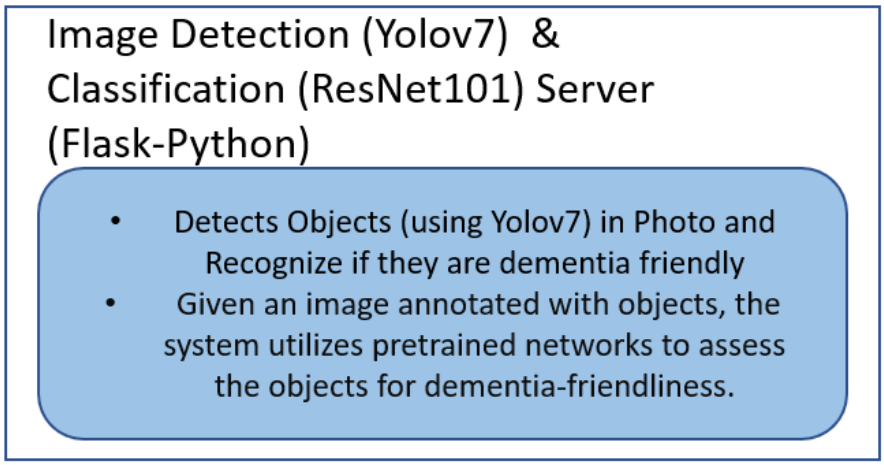

Figure 4.15: Image Detection Component

4.2.5.1 Dataset

The dataset was built from 1.3k images of prayer rugs and 2.5k images of carpets, as referenced from [42], with an additional 100 carpet photos obtained through a scraping script. It should be noted that the dataset received extensive refinement for the purposes of this thesis. This refinement was critical to remove duplicate and low quality images. Following this cleansing process, the dataset was annotated using the Darknet format [43] using LabelImg [44]. LabelImg is a graphical image annotation tool utilized for preparing images for object detection model training. It should be highlighted that the user-friendly interface of LabelImg significantly streamlined the annotation process.

The Darknet framework, utilized in the YOLO (You Only Look Once) object detection system, adopts a specific annotation format for labeling objects in images. This format is essential for the effective training of the model to recognize and locate objects [43].

1. **File Correspondence**: Each image in the training set is paired with a corresponding annotation file. This file is named identically to the image but with a `.txt` extension. For example, the annotations for `image.jpg` are in `image.txt`.
2. **Line Structure in Annotation Files**: Every line in the text file corresponds to a single object in the image and contains several numerical values:
   - **Class ID**: The first number represents the class of the object, starting from 0, aligning with the order of classes in the model's class list.
   - **Normalized Bounding Box Coordinates**: The subsequent four numbers represent the bounding box's normalized position and size relative to the image's dimensions, which include:
     - **Center X and Y**: Proportions of the bounding box's center coordinates to the image's width and height.
     - **Width and Height**: The bounding box's width and height as fractions of the image's dimensions.
3. **Normalization Explained**: All these numbers are fractions of the picture's size. So, they're always between 0 and 1, no matter how big the actual picture is.
4. **Example Annotation**: In an 800x600 pixel image, an object classified as number 2 (e.g., a bicycle), centrally located with a 200x150 pixel bounding box, is annotated in `image.txt` as:

   ```
   2 0.5 0.5 0.25 0.25
   ```

   This denotes the bounding box's normalized center, width, and height.
5. **Boxes are Straight**: This format assumes the boxes around objects are straight and not tilted.

It should be noted that the dataset needed to be modified to meet the ResNet model's input requirements, which differ from those of the YOLOv7 model. A script was developed to reformat the dataset appropriately and align it with the input requirements of the ResNet-101 model.

Object Detection with YOLOv7: In this research, YOLOv7 is applied to detect and evaluate objects within environments designed for individuals with dementia. This method categorizes a wide array of items, including carpets, mirrors, and tables, determining if they are friendly or not for dementia-supportive settings. An example of the dataset for the aforementioned object classes would be:

- carpet_good: Carpets that meet the criteria for being dementia-friendly.

- carpet_bad: Carpets that do not meet the criteria and could pose a risk or confusion.
- mirror_good: Mirrors that are safe and suitable for individuals with dementia.
- mirror_bad: Mirrors that could potentially cause distress or confusion.
- table_good: Tables that are appropriate and secure for use by individuals with dementia.
- table_bad: Tables that are deemed unsafe or unsuitable.

This categorical breakdown allows the YOLOv7 model to not only detect these objects within various settings but also to classify them according to their suitability for a dementia-friendly environment.

Object Classification using ResNet-101: Following successful object detection, facilitated by user annotations, the system advances to the classification stage, employing the ResNet-101 architecture. For each detected object, its coordinates and assigned class category guide the selection of a dedicated pre-trained neural network for that specific class. These individual networks have been trained to classify objects as 'good' or 'bad' within their respective categories. For instance:

- For any object identified as a 'carpet', the corresponding 'carpet classification network' is employed. This specialized network contains the binary classes 'good' and 'bad' to evaluate the dementia-friendly status of the carpet.
- Similarly, if pre-trained classification networks for mirrors or doors are available, they would assess these items for dementia-friendliness, categorizing them as either 'good' or 'bad'. While the 'mirror classification network' and 'door classification network' were not pre-trained as part of this thesis, the architecture of the system is strategically designed for easy integration of these networks once they are developed. If these networks were pre-trained, users could label objects they've identified, like mirrors or tables, within the system's interface and the corresponding classification network would be activated to evaluate the labeled object, determining if it's 'good' or 'bad' for dementia-friendliness.

This detailed method guarantees that every object is assessed by a neural network adjusted for its specific features, improving the accuracy of evaluations regarding its suitability for dementia-friendly environments.

4.2.5.2 Folder Structure

The folder structure for the image detection component of the project is designed for modularity and scalability and consists of the following directories:

- **thesis_code**: This is the root folder for the custom code that extends the functionality of YOLOv7 for the specific needs of the project.
  - **common**: Contains shared resources used across different parts of the project, such as middleware and error handling.

- **weights_classification_networks**: Stores the weight files for each classification network, enabling individualized assessment of object classes.
- **features**: Houses specific features of the application.
  - **detection**: Currently the only implemented feature, with the following structure:
    - **config.py**: Contains constant variables for configuration settings.
    - **utils.py**: Provides utility functions supporting detection operations.
    - **routes.py**: Defines endpoints for the image detection API.
    - **decorators.py**: Contains decorators for route endpoints, facilitating tasks like authentication or validations.
    - **services**: Subdirectory containing implementations for the two different classification option (detection using yolov7 and Dynamic classification using Resnet101)

- **global_utils**: Top-level folder for code common across the entire application.

The application's entry point is located in the `app.py` file, which sets up and starts the service.

4.2.6 Performance Metrics

Metrics are defined as the standards of measurement by which the effectiveness, progress, and accuracy of a model or algorithm can be evaluated. The following metrics are commonly used in classification models.

Confusion Matrix A Confusion Matrix is a table that describes the performance of a classification model on a set of test data for which the true values are known. It allows you to visualize the accuracy of the model in terms of correct and incorrect predictions. Confusion Matrix Elements are below:

- **True Positives (TP):** These are cases in which the model correctly predicts the positive class.
- **True Negatives (TN):** These are cases in which the model correctly predicts the negative class.
- **False Positives (FP) or Type I Error:** These are cases in which the model incorrectly predicts the positive class.
- **False Negatives (FN) or Type II Error** These are cases in which the model incorrectly predicts the negative class.

Accuracy Accuracy measures the overall correctness of the model and is calculated by the formula

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Precision (Positive Predictive Value, PPV): Precision assesses the proportion of positive identifications that were actually correct and is defined as:

$$\text{Precision (PPV)} = \frac{TP}{TP + FP} \tag{4.2}$$

Recall (True Positive Rate, TPR) or Sensitivity: Recall measures the proportion of actual positives that were correctly identified and is defined as:

$$\text{Recall (Sensitivity, TPR)} = \frac{TP}{TP + FN} \tag{4.3}$$

Specificity (True Negative Rate, TNR) Specificity measures the proportion of actual negatives that were correctly identified as such:

$$\text{Specificity (TNR)} = \frac{TN}{TN + FP} \tag{4.4}$$

F1 Score: The F1 Score is the harmonic mean of precision and recall, providing a balance between the two:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.5}$$

Mean Average Precision (MAP): Mean Average Precision is a measure used to evaluate the quality across multiple query terms. It averages the precision scores after each relevant document is retrieved, and then averages this over all query terms. It is particularly useful in information retrieval contexts where the order of retrieval is significant:

$$\text{MAP} = \frac{1}{Q} \sum_{q=1}^{Q} \frac{1}{m_q} \sum_{k=1}^{m_q} \text{Precision@k} \tag{4.6}$$

Here, Q represents the total number of queries, m_q is the number of relevant documents for query q, and Precision@k is the precision at cutoff k, i.e., considering the top k results.

4.3 Containerization Strategy

This chapter delves into the development and containerization strategy implemented for the dementia-friendly image detection tool, with a specific focus on facilitating future enhancements with microservices. Containerization is very important in the development process and provides notable benefits such as portability and reproducibility of the software environment. Before delving deeper into the implemented strategy, it's crucial to acknowledge the importance of wise and effective modularization in software projects. This principle separates different concerns into smaller, more manageable modules, thereby enhancing maintainability and scalability. The

optimal modularization, particularly when used along with techniques such as containerization, serves as the foundation for transitioning into microservices architecture in the future that can offer the ultimate maintainability and scalability. Given that microservices require a lot of boilerplate configuration, this thesis prioritizes modularity and containerization to create a robust groundwork for potential microservices integration. The current strategy was inspired by a Udemy instructor Stephen Grider [41] and the containerization platform used is Docker.

4.3.1 Modularization Of Components and DockerFiles

Modularization of components of the proposed tool provides separation of concerns and offer highly maintanibility and easier extension of the current tool. In this thesis the project was modularized into 4 distinct components, each one responsible for specific functionalities. As mentioned in Section 4.1, these components modules are the following:

1. **Client (React Web Application)**
2. **Authentication Service (Node.js Application)**
3. **Image File Server (Node.js Application)**
4. **Image Detection Server (Flask Application)**

Dockerfiles provide instructions for building Docker images, which encapsulate the application and its dependencies into portable containers. These Dockerfiles are tailored to the specific requirements of each component and for the first three applications (Client, Authentication Service, and Image File Server) are nearly identical since they rely on the Node base image. Similarly, the Dockerfile for the Image Detection Server, which operates with Flask and Python, follows a similar structure adapted for Python dependencies. The typical structure of these Dockerfiles includes several key steps that are essential for the subsequent building and deployment of these services, as presented below:

- **Selecting a Base Image:** Begins with choosing an appropriate base image, like Node for JavaScript (Node) applications or a Python image for Flask applications. This image serves as the foundation upon which the container is built.
- **Setting a Working Directory:** Establishes a directory inside the container where all the commands are run.
- **Copying Dependency Files:** Moves the package.json (for Node.js applications) or requirements.txt (for Python applications) to the container to install dependencies.
- **Installing Dependencies:** Runs npm install (for the js applications) or pip install (for python applications) to install the required packages listed in the dependency files.
- **Copying Application Files:** Transfers the rest of the application's source code into the container.

- **Specifying the Start Command:** Defines the command to start the application, such as npm start or a Flask application run command.

These steps are key in creating Docker images for each part of the system. The commands for building and deploying these Docker images include:

```
docker build -t <image_name> .
```

And running the container, mapping necessary ports:

```
docker run -p <host_port>:<container_port> <image_name>
```

Here, `<image_name>` should be replaced with the desired name for the Docker image. The first command builds the Docker image using the Dockerfile located in the current directory (`.`), while the second command runs the Docker image mapping ports between the host and container (`-p <host_port>: <container_port>`), and specifying the name of the Docker image to run.

Modularizing every code component and utilizing Docker for building and deployment is a very effective architectural approach. This approach guarantees maintainability while simplifying development. It also lays the groundwork for developing scalable systems and guaranteeing the seamless integration of upcoming updates.

4.3.2 Deployment

After discussing the Dockerfiles for individual components, it's essential to understand how these components are coordinated and deployed as a cohesive system. Before specifying how these components are deployed together using configuration files written in (YAML), the layered architecture the containerized applications are structured around will be presented.

The deployment architecture for containerized applications is built on a four-layer stack (Fig. 4.16): starting with the **physical or virtualized hardware** where everything runs, followed by the **host operating system** that provides the software foundation. **Docker** sits above this, creating isolated containers for application deployment and management. At the top, **Docker Compose** orchestrates these containers, ensuring seamless operation across the application's 5 key services. As illustrated in Fig. 4.16 the 5 key services are:

1. **Client (React Web Application)** This service represents the React client application.
2. **Authentication Service (Node.js Application)** This service represents the authentication server and depends on the mongo service.
3. **Image File Server (Node.js Application)**: This service handles image file storage and retrieval. It also depends on the mongo service.
4. **Image Detection/Classification Server (Flask Application):** This service handles both image detection and classification.
5. **MongoDb** This service utilizes the official MongoDB image and acts as the database backend for the application. It creates a volume named mongo-data

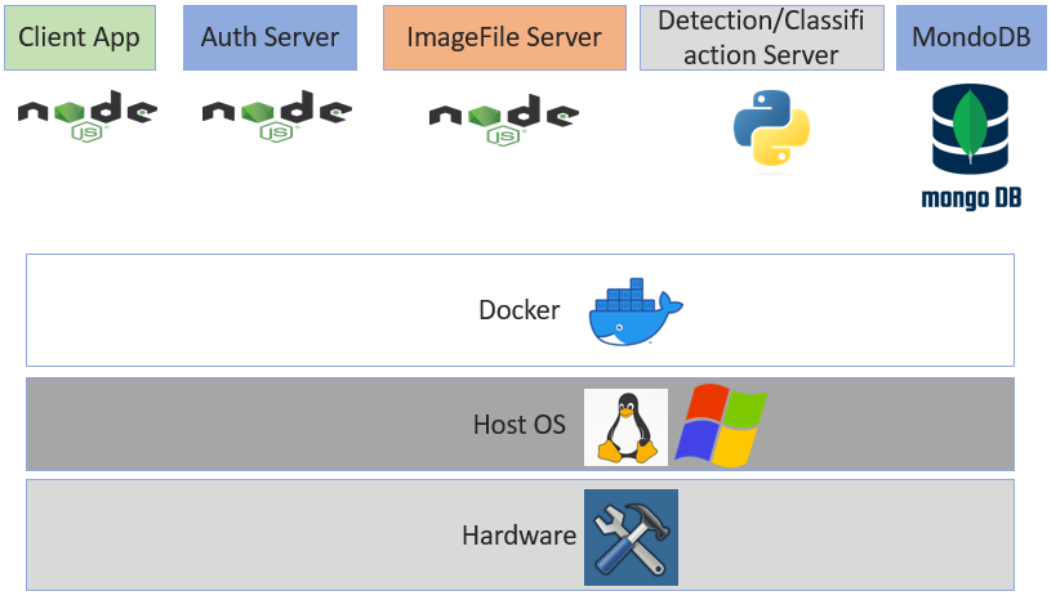

Figure 4.16: Layered Architecture ending to the containerized applications/services used at the current thesis

to persist database data.

4.3.2.1 The Role of Docker Compose

The Docker Compose file is central to the deployment process, specifying the services, networks, and volumes necessary for the application. Its key components include:

- **version**: Specifies the version of the Docker Compose file format being used.
- **services**: Defines the services used.
- **container_name**: Specifies the name of the container for each service.
- **build**: Specifies the build context for each service, indicating the directory containing the Dockerfile to build the image from.
- **ports**: Maps ports between the host and container.
- **environment**: Specifies environment variables to be passed to the containers.
- **depends_on**: Specifies dependencies between services, ensuring that services are started in the correct order.
- **image**: Specifies the Docker image to be used.
- **volumes**: Defines volumes to be mounted for persistent data storage.

In this subsection, the importance of modularization in the project is acknowledged, demonstrating its facilitation of deployment's architecture. This modular approach, integrated within Dockerfiles and managed by Docker Compose, promotes code

reusability and facilitates future enhancements and microservices integration, all while making deployment processes more efficient.

5

Experiments & Results

This chapter details the experiments conducted for the detection and classification of objects using the dataset compiled for this study. The dataset comprises 1,300 images of prayer rugs and 2,500 images of carpets, as referenced from [42], augmented with an additional 100 carpet photos obtained through a specialized scraping script. The total dataset sizes are 100 MB for the prayer rugs, 56 MB for the carpets, and approximately 1 MB for the scraped images. The dataset was partitioned into training, validation, and test sets with an 80/10/10 split. Throughout the thesis, the dataset underwent refinement to remove potential duplicates and images of low analysis.

Object Detection with YOLOv7: Initially, the results of object detection using YOLOv7 are presented and discussed. This phase aims to evaluates YOLOv7's ability to detect and categorize multiple carpets within a single image as either 'good' or 'bad'. The classification of carpets as 'good' or 'bad' is determined by their suitability for environments designed to be friendly for individuals with dementia, focusing on aspects that might affect individuals' psychology or navigation. Performance is measured using metrics detailed in previous chapters, including precision, recall, F1 score, and confusion matrix in order to assess YOLOv7's model effectiveness.

Object Classification with ResNet-101: Following the object detection results, the classification outcomes achieved using ResNet-101 are presented. Contrary to YOLOv7's object-focused approach, ResNet-101 classifies the entire image as depicting a 'good' or 'bad' carpet. The model performance is demonstrated through performance metrics such as precision, recall, F1 score, and visual representations in graphs, providing insight into ResNet-101's capability to accurately classify images based on their suitability for dementia-friendly environments.

Web Application Interface: Finally, the web application developed to interface with the detection and classification systems is introduced. This section will include

screenshots and descriptions of the application's functionality, illustrating its use with specific cases related to the activity diagrams presented in the previous chapter. This part aims to demonstrate how the theoretical and technical components of the thesis are applied in a practical environment.

5.1 Results for the Object Detection/Classification

5.1.1 Object Detection with Yolov7

As aforementioned the initial dataset underwent refinement resulting in a collection of 2,292 images of carpets, categorized as 'good' or 'bad'. This refined dataset was then divided using an 80/10/10 split for training, validation, and testing purposes, respectively.

To address the task of carpet detection, distinguishing between 'good' and 'bad' categories, the yolov7 model was retrained using the refined dataset. The key elements of this training configuration are outlined below:

- **Batch Size:** A batch size of 8 was used for training to optimize the balance between memory usage and model performance.
- **Device:** Training was conducted on a single GPU to ensure efficient computation.
- **Image Size:** All images were resized to 640x640 pixels. This standardization was necessary to maintain consistency in input size for the model.
- **Model Configuration:** The YOLOv7 model was adapted for carpet detection, with a focus on identifying the 'good' and 'bad' classes. This adaptation included tuning the model to recognize the specific characteristics of our dataset.
- **Pretrained Weights:** The Yolo's v7 pretrained weights available from the official YOLOv7 website were exploited.
- **Hyperparameters:** The Custom hyperparameters specific to YOLOv7 were employed for the first result/
- **Epochs:** The dataset was retrained for 300 epochs to ensure the model will adequately learn from the dataset.

The performance of the model for the above configuration is assessed using the set of metrics presented in previous chapter. These metrics are crucial for understanding the models' precision, recall, and accuracy in various scenarios. The results are presented with metrics in the curves depicted in Figures (Fig. 5.1-5.4)

Recall Curve: As depicted in Fig. 5.1, a recall of 0.94 at a confidence threshold of 0 indicates that the model successfully identifies 94% of all true positive cases across carpet classifications (good_carpet and bad_carpet) , without requiring any minimum level of confidence in its predictions. This high recall suggests the model's strong capability in detecting true positives. However, the absence of a confidence threshold may also lead to a significant number of false positives. For instance, at a more typical confidence threshold of 0.25, the recall dramatically drops to 0.5,

indicating that the model's effectiveness substantially decreases as it becomes more selective in its predictions. This suggests that the model does not perform as well when it needs to be more certain about its predictions, implying that precision and overall accuracy may be compromised for higher level of confidence

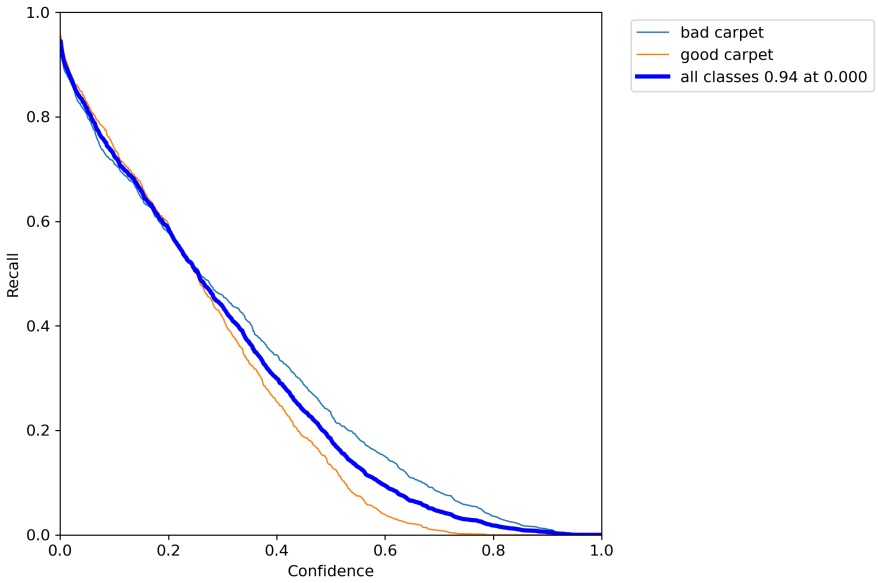

Figure 5.1: Recall Curve

Precision Curve As Fig. 5.2 demonstrates a complementary metric of the model's performance. Achieving a precision of 1 at a confidence threshold of 0.953 means that when the model is highly confident (nearly 95.3% sure), it makes predictions with high accuracy, not identifying any false positives at this high level of confidence. This indicates that the model is extremely reliable in its most confident predictions. However, this remarkable precision underscores a trade-off: as the model becomes more selective (choosing only those predictions it's nearly certain about), it may overlook some true positives. To identify the balance between precision and recall effectively, the F1 score curve have to be examined. The F1 score combines precision and recall into a single metric and it allows to identify the optimal confidence threshold that maximizes both metrics, ensuring the model's overall performance is finely tuned.

F1 Score Curve The F1 score, as shown in Fig. 5.3, peaks at 0.57 with the confidence threshold set to 0.197, marking the best balance the model achieves between recall and precision. This peak represents a balanced average of the model's ability to correctly identify true positives and its precision to ensure those identifications are accurate. In many contexts, an F1 score around the mid-point of 0.6 maybe be quite acceptable, especially in less critical applications where the consequences of a mistake are minimal. Given the importance of precise carpet classification in creating safe environments for individuals with dementia, our model's F1 score indicates there's significant room for improvement to meet the necessary high accuracy and reliability.

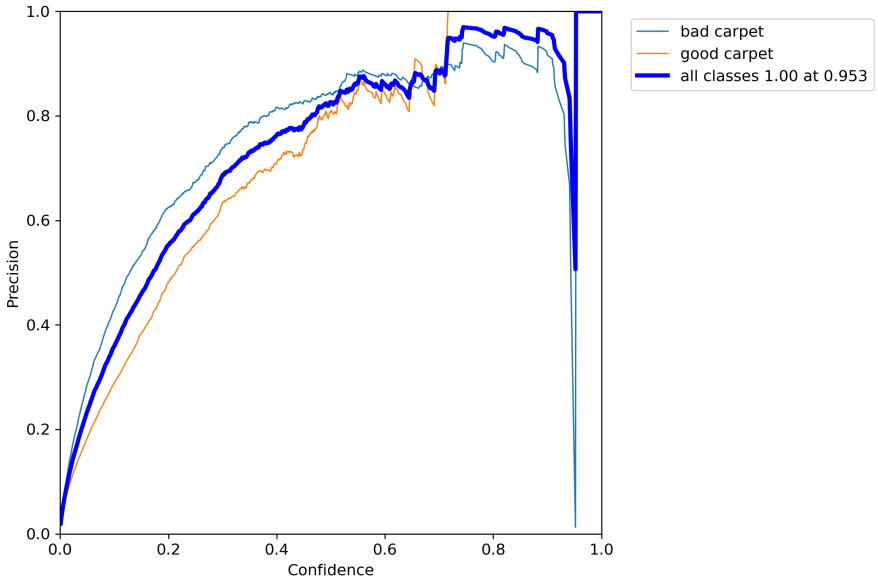

Figure 5.2: P curve

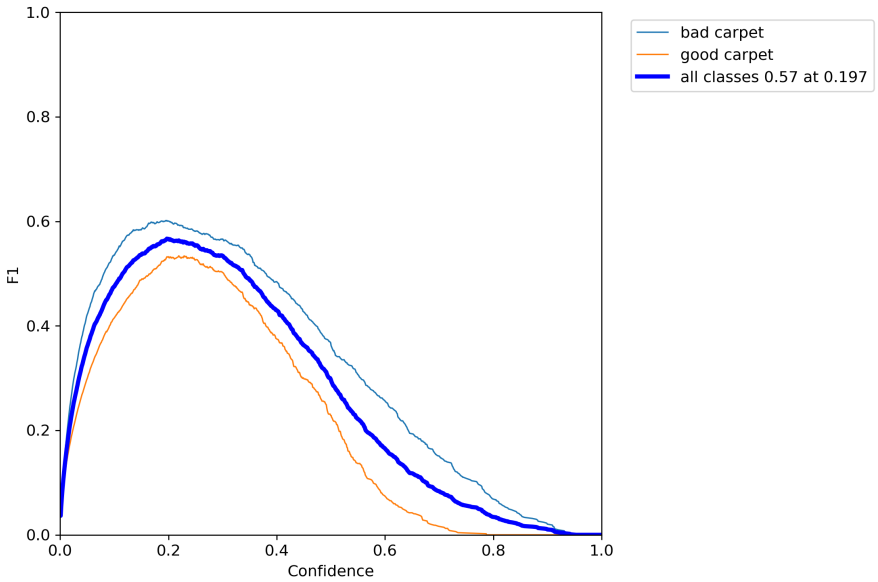

Figure 5.3: F1 curve

Precision-Recall Curve Analysis The precision-recall curve shown in Fig. 5.3 reveals a mean Average Precision (mAP) of 0.547 at an IoU threshold of 0.5, reflecting the model's average detection performance across all classes. This mAP value represents a fair level of model accuracy in distinguishing between correct (true positive) and incorrect (false positive) carpet classifications. When model becomes more sensitive by increasing recall, it's likely to capture more true positives but also to mistakenly classify more non-carpet as carpets (false positives). Given the critical nature of our task, which demands high precision for the safety and comfort of individuals with dementia, the current mAP implies that a notable improvement is needed to make the model detect decently.

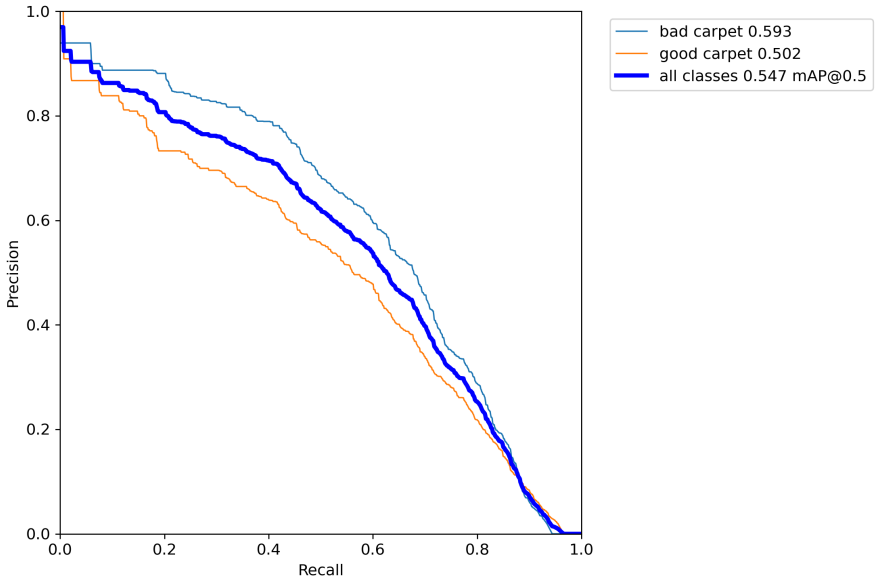

Figure 5.4: PR curve

Prior to summarizing, it is important to highlight the performance of the detection model that utilized YOLOv7. The Figs. 5.5 - 5.6 showcase a comparison between real labels and the predictions made by the model for a test batch of 16 images, including the model's confidence in classifying the carpets. These images illustrate the challenges faced by the YOLOv7 model in accurately detecting and classifying carpets as either good or bad. The unsatisfactory outcomes of the model, underscore the detection model's limitations and pave the way for an improved strategy that focuses solely on classification, as discussed in the subsequent sections.

To sum up the outcomes, the results are not particularly strong, and this could stem from the limitations within our dataset, which is not ideally balanced or of the highest quality or there are still duplicates. This challenge led to focus less on fine-tuning the model's hyperparameters, like the learning rate or the number of training cycles (epochs), as doing so may not substantially improve the results given the dataset issues. The confusion matrix depicted in Fig. 5.7 provides an insight into how the model has performed on the test set. The results are not satisfactory and the performance of the model is not considered good. It appears the model often confuses between classes, indicating a need for improvement in its ability to distinguish carpets within the provided dataset.

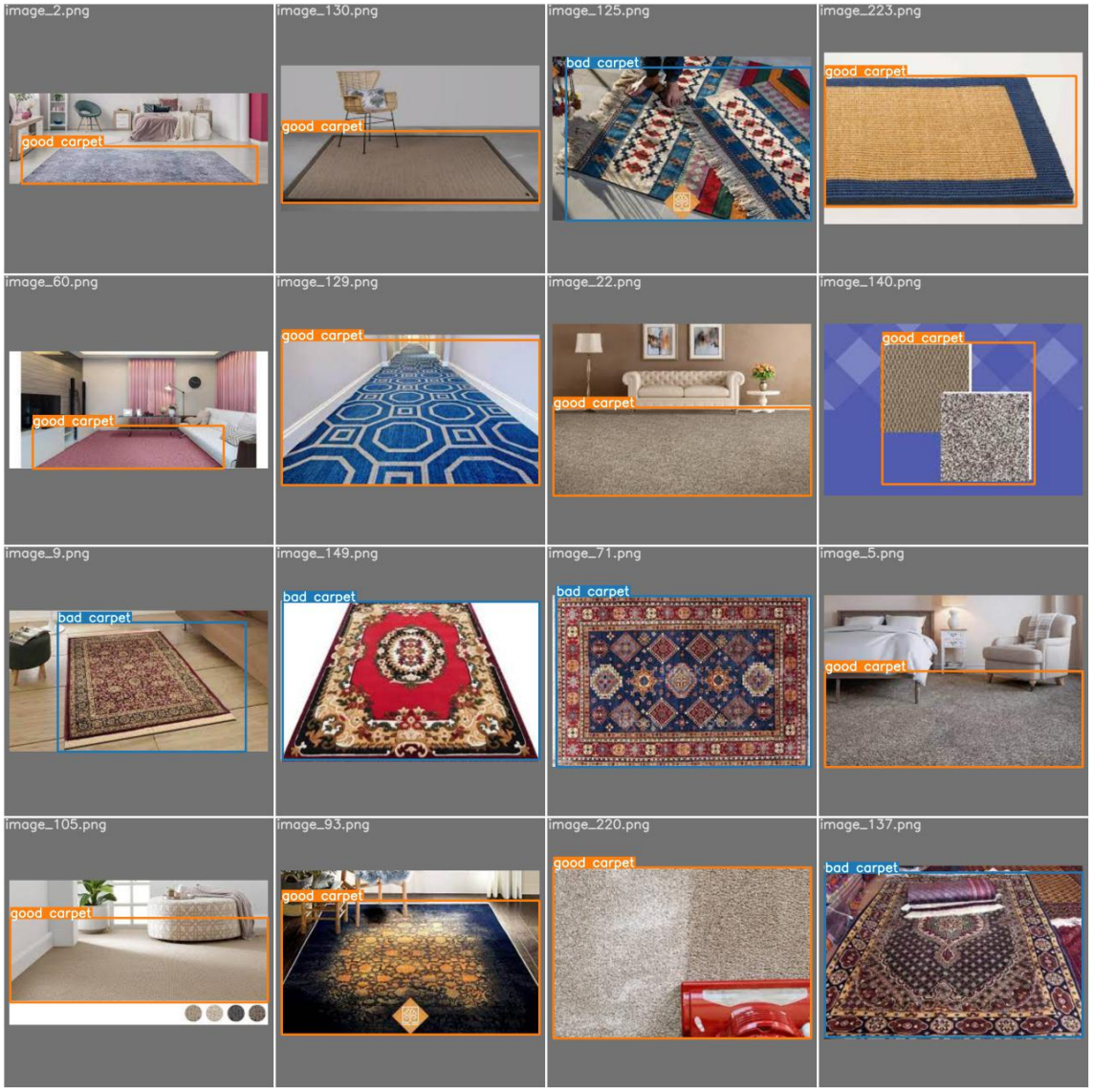

Figure 5.5: Example of real labels for a test batch of carpets.

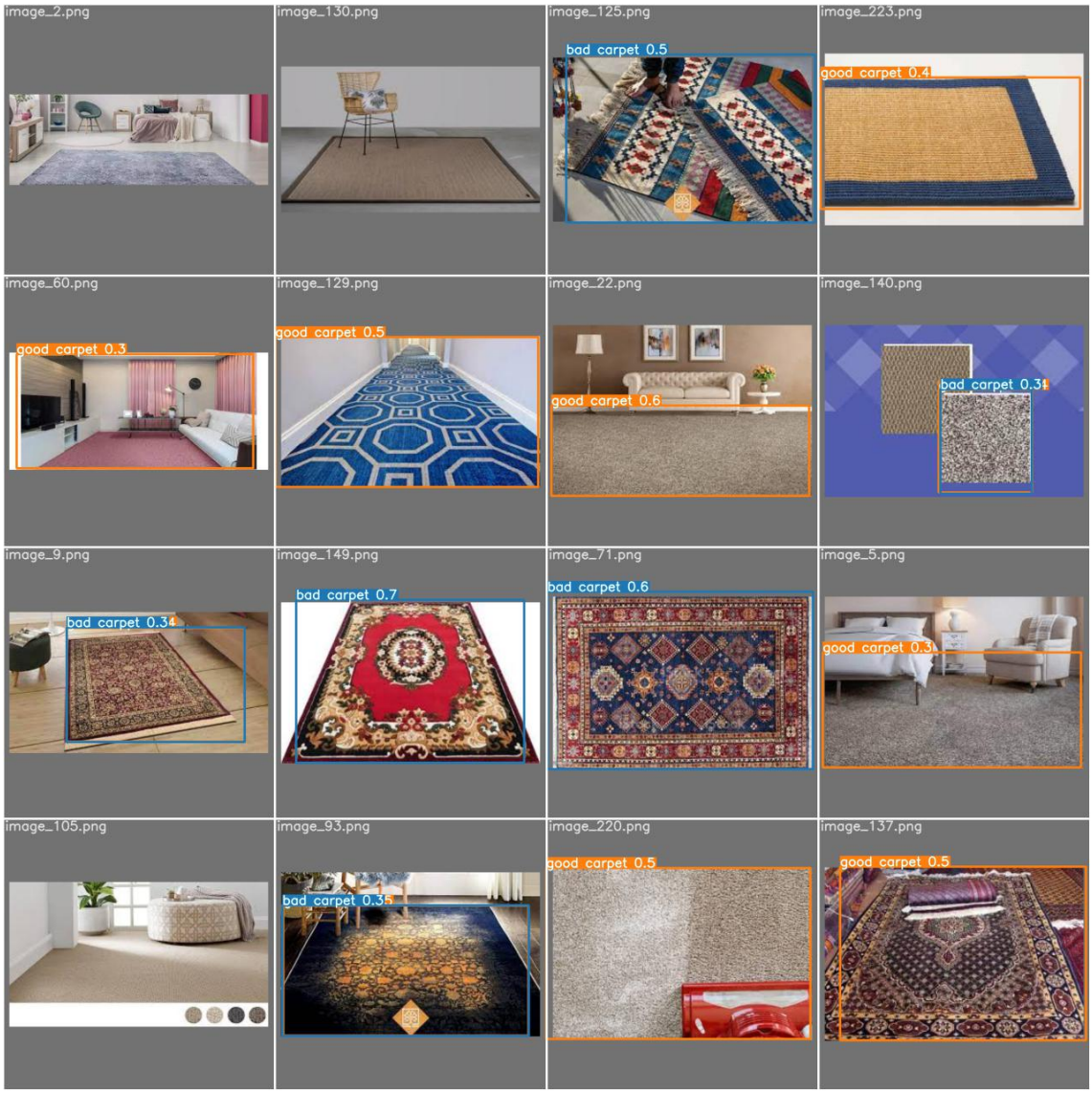

Figure 5.6: Detection model predictions for the same test batch, showing classification confidence.

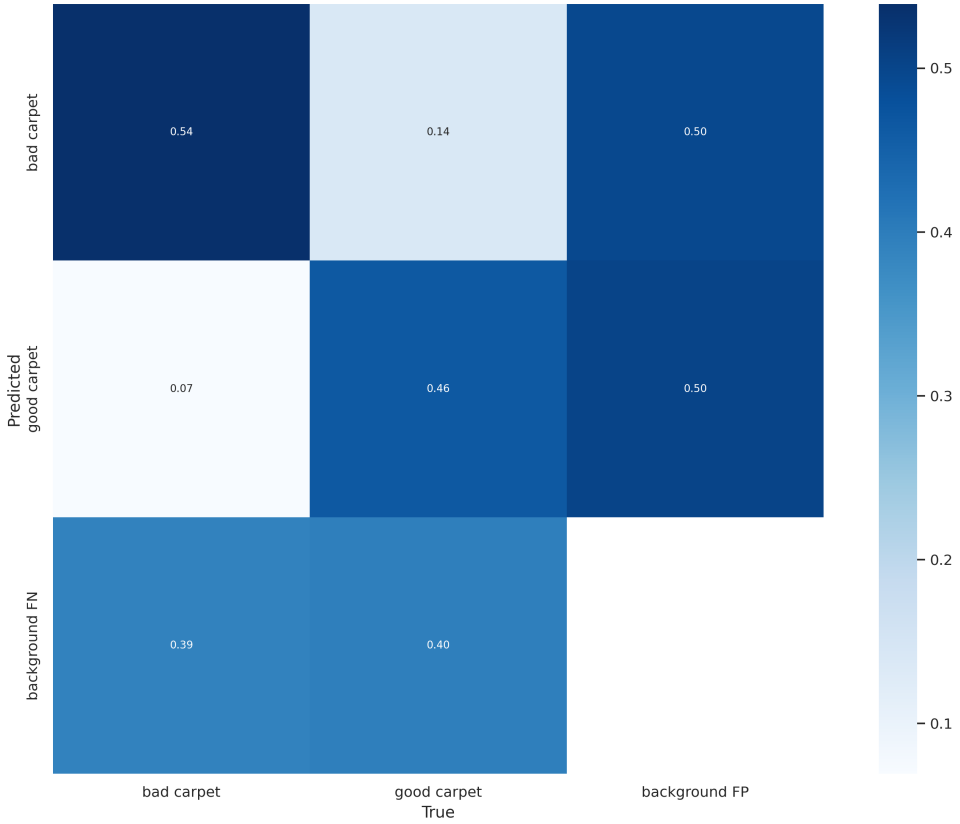

Figure 5.7: F1 curve

5.1.2 Object Classification

Dataset Adaptation and Model Training Despite the dataset's lack in reliability and in order to accurately identify good or bad carpets, a ResNet-based classification model was trained to compare the results with the yolov7 model. This model was tasked with a simpler objective: to classify images of carpets as 'good' or 'bad', given an image that contains a single carpet. This method aims to enhance accuracy by reducing the complexity of the task, because rather than detecting and classifying multiple objects within an image, the model focuses solely on the carpet itself.

It should be noted that the dataset needed to be modified to meet the ResNet model's input requirements, which differ from those of the YOLOv7 model. This dataset consisted of 2,592 carpet images. Unlike the straightforward split used previously, another approach to partition the data was employed: k-fold cross-validation was utilized, specifically with 5 folds, to enhance the robustness and reliability of our model evaluation. This means that 90% of the dataset was used for training and validation while the remaining 10% of the dataset was reserved exclusively for testing. This method was particularly beneficial due to the nature of the available dataset, which still contained a larger number of flawed-images even after the multiple refinements. The dataset contained a greater number of images compared to those utilized in the detection dataset (2292 images of carpets) because the script extracted multiple carpets from a single image based on their annotations (if there were more than one). To further enhance the dataset's quality, an additional refinement step was implemented to eliminate duplicates and poor-quality images.

The dataset underwent several transformations, such as random rotation and horizontal flipping, to boost the model's performance. The training utilized a batch size of 32 and employed the SGD optimizer with a learning rate of 0.001 and momentum of 0.9. Furthermore, the ResNet model was modified by removing its last layer and retraining it with the refined carpet dataset. To avoid overfitting, the model's parameters were saved only if the validation loss improved. The initial experiment spanned three epochs.

Performance Analysis and Results : The performance of the ResNet-based model was firstly evaluated in the validation set using cross validation with 5 folds as mentioned. During the cross-validation process, the best model for each fold was determined by measuring the loss and accuracy at every epoch. The models that performed best on the training set were then evaluated on the validation set, and these metrics were averaged to find the mean. As illustrated in Figure 5.8, all metrics hovered around 0.90. This indicates that the model's ability to classify a carpet as good or bad has remarkably improved compared to the previous approach, where YOLOv7 was used for detection to identify a carpet in an image and directly classify it as good or bad.

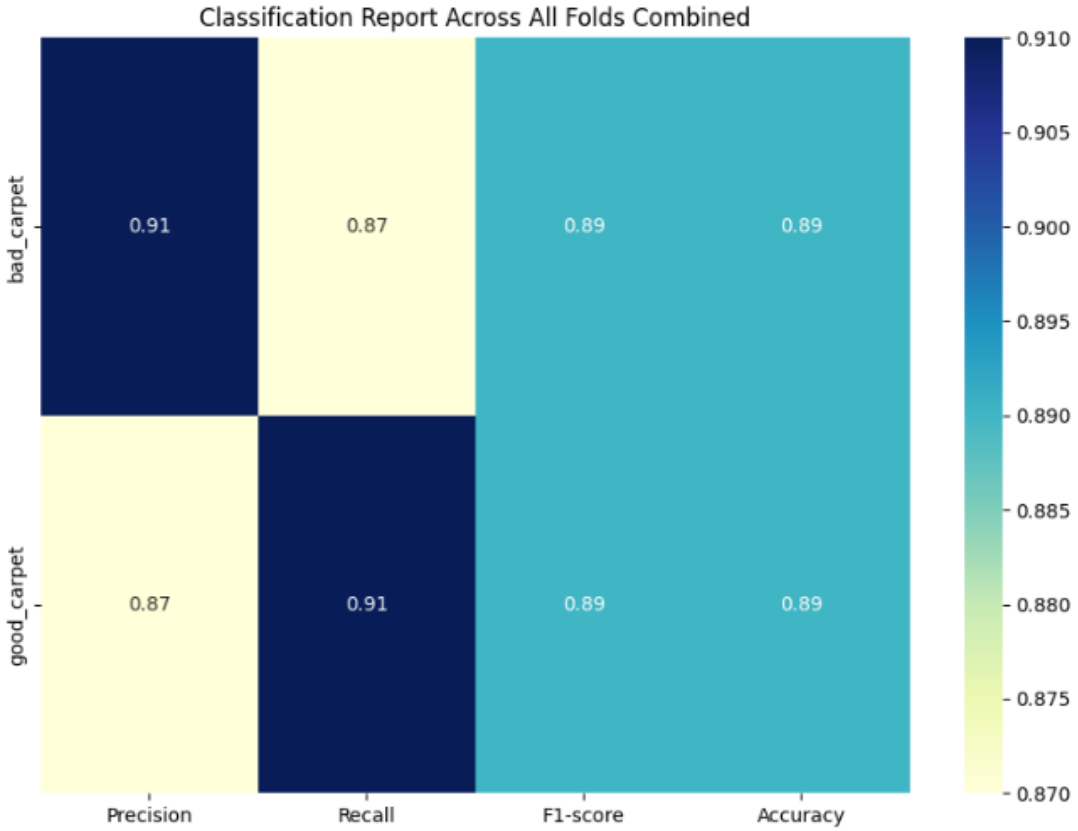

Figure 5.8: The mean of the performance metrics using the best model for each fold after cross-validation.

The use of cross-validation was necessitated because of the limitations of the dataset, which includes a lot of low quality images. Even after a thorough refinement process, a traditional split into training, validation, and test sets could introduce significant drawbacks, with low-quality photos being unevenly distributed among the sets.

This uneven distribution could potentially bias the model's performance evaluation. Cross-validation allowed for a more reliably assessment because it ensured that every part of the dataset was used for both training and validation across different folds. Despite the dataset's challenges, validation accuracy for each fold consistently remained above 85%. As illustrated in Fig. 5.8, the right column of the table within this image visualizes the average accuracy for all folds, demonstrating enhanced model performance across varying conditions.

In this thesis, a specific method was followed to assess carpets as being dementia-friendly or not. The method utilized cross-validation with five folds, where each fold underwent three epochs of training. The model that demonstrated the best performance in terms of accuracy and the balance between precision and recall across these folds was selected and subsequently evaluated using a test set comprised of unseen data. A significant difference between the validation metrics and a lower performance on the test set (despite better performance on the validation set) would suggest that the test set may contain a higher proportion of low-quality images.

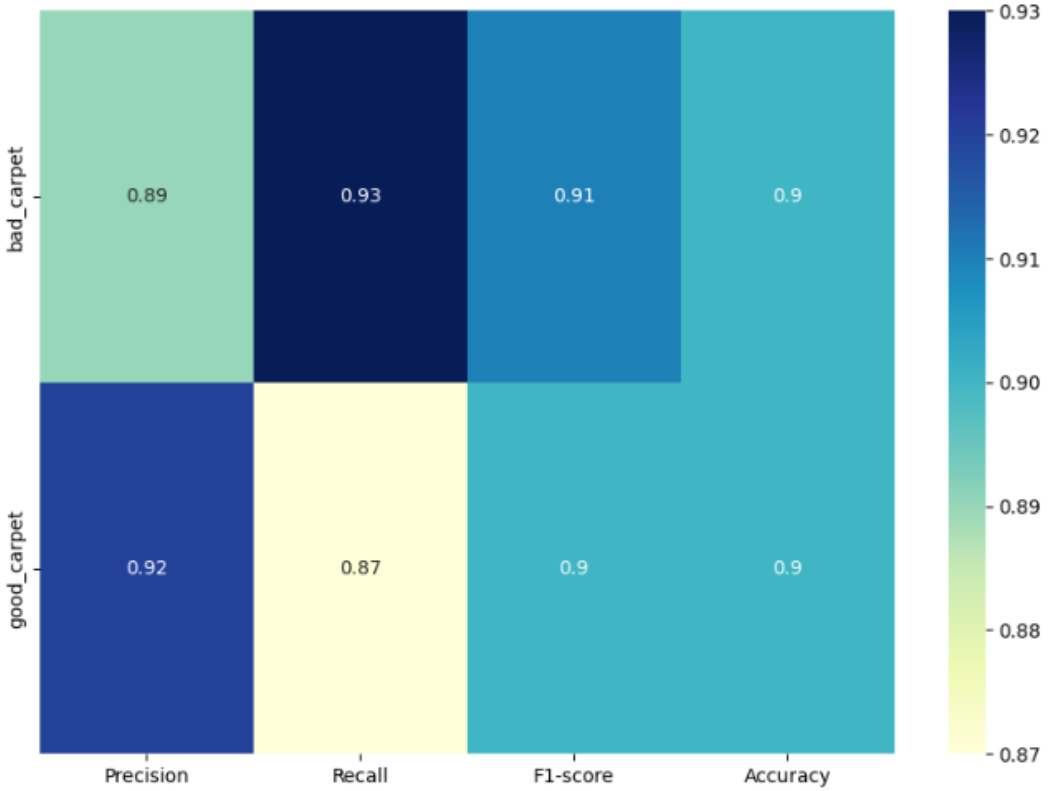

Figure 5.9: Performance metrics of the test set using the best performing model selected from the cross-validation folds over all epochs.

The outcome of this evaluation process is depicted in Fig. 5.9. Remarkably, the metrics for the test set demonstrated even better performance compared to the mean metrics of the evaluation set. Specifically, for the classification of bad carpets, the model achieved precision, recall, and F1 score metrics of 0.89, 0.93, and 0.91, respectively. For good carpets, the metrics were equally impressive, with precision, recall, and F1 score values of 0.92, 0.87, and 0.90, respectively. These metrics, derived from the confusion matrix illustrated in Fig. 5.10, highlighting the model's

exceptional ability to distinguish between good and bad carpets accurately. An overall accuracy of 90% further proves the model's robustness.

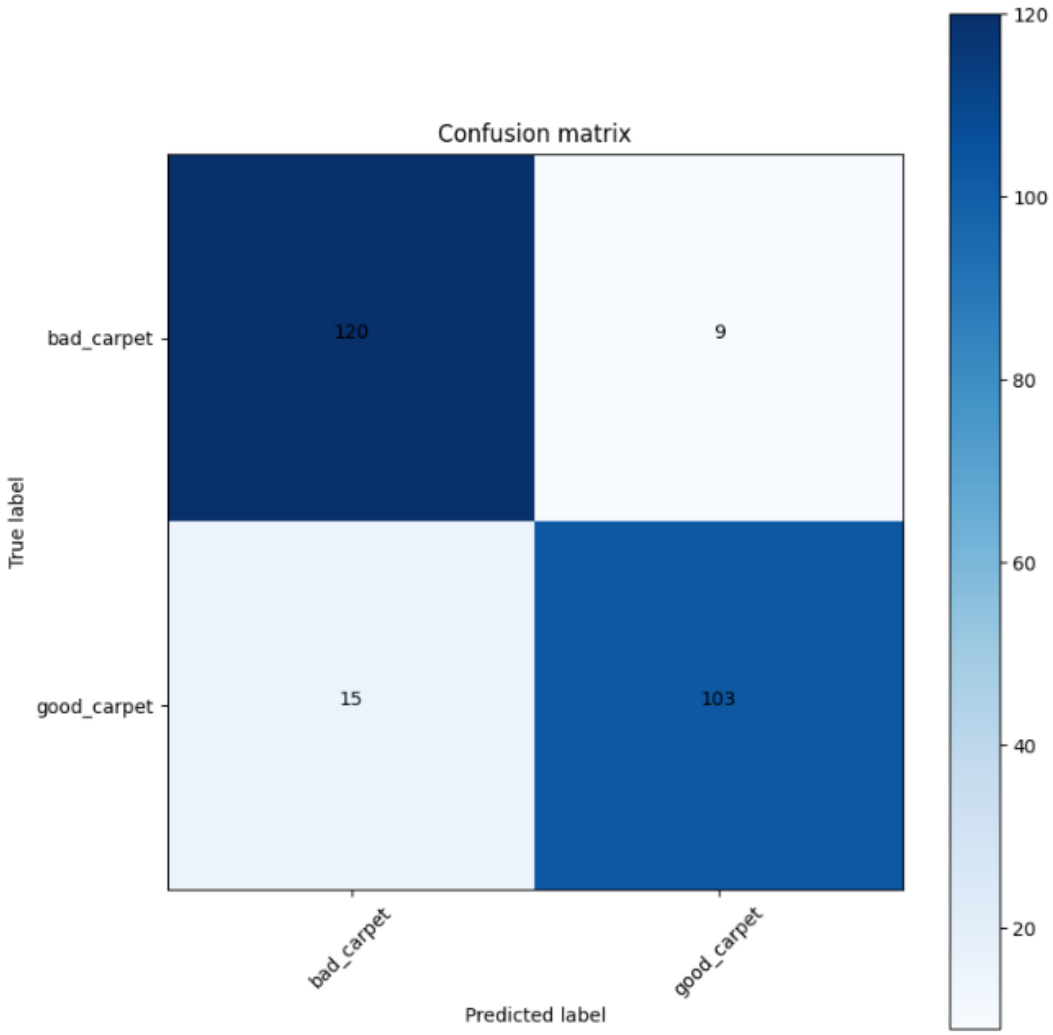

Figure 5.10: Confusion matrix showcasing the test set outcomes using the best performing model from all epochs across the cross-validation folds.

The model's high precision and recall significantly improve its predictions on dementia-friendly carpets. High precision ensures that most carpets labeled as good are accurately identified, minimizing false positives. High recall means the model effectively recognizes most actual dementia-friendly carpets, with few false negatives. These strengths confirm the model's reliability in classifying carpets accurately.

In comparison, this model outperforms previous attempts that used YOLOv7 for merely detecting carpets in images. Unlike YOLOv7, which had difficulties with simultaneous detection and classification of carpets as dementia-friendly, the current model excels in classifying carpets as dementia-friendly. This marks a notable advance, demonstrating the current method's precision and effectiveness in providing a refined assessment of carpets for dementia-friendliness.

Training vs. Validation Accuracy Graph Discussion Figure 5.11 shows the Training vs. Validation Accuracy graph, depicting the average model accuracy across

five folds of cross-validation over three epochs. The mean accuracy for each epoch (E) across the five folds (K) is determined by Eq. 5.1. Initially, the training accuracy is lower, yet it quickly climbs, demonstrating consistent improvement. This suggests the model is successfully learning from the training dataset. In contrast, the validation accuracy starts higher and rises in parallel with the training accuracy, indicating the model's initial effectiveness with unseen data. However, a slight decline after the second epoch might signal that the model is beginning to overfit. This overfitting may imply that the model is possibly learn irrelevant details from the training data, which don't help it perform better on the validation set. It's important to mention that across all five folds, each fold exhibited improved performance in the second epoch compared to the third. Ideally, exploring more epochs could provide further insights into the model's learning curve and potential for even better accuracy. However, this exploration was constrained by the limited resources available for the training process.

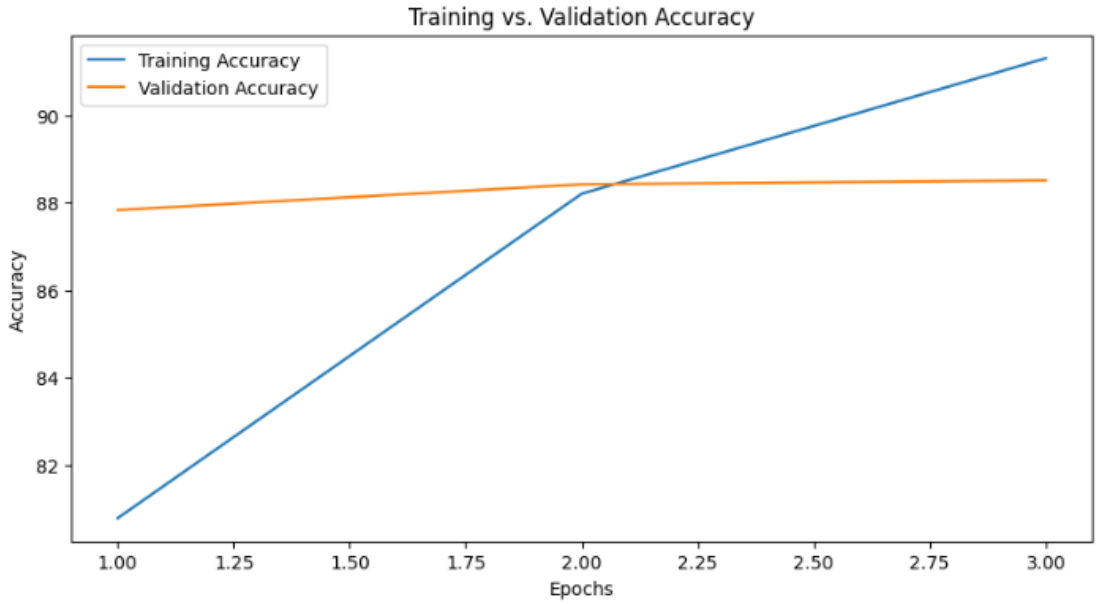

Figure 5.11: Comparison of average training and validation accuracy across epochs, based on five-fold cross-validation.

$$\text{Average Accuracy at Epoch } E = \frac{1}{K}\sum_{k=1}^{K}\text{Accuracy}_{E,k} \tag{5.1}$$

$$\text{Average Loss at Epoch } E = \frac{1}{K}\sum_{k=1}^{K}\text{Loss}_{E,k} \tag{5.2}$$

Training vs. Validation Loss Graph Discussion Similarly, the Training vs. Validation Loss graph (Fig. 5.12) reflects the learning process, with the mean loss for both training and validation across the five folds. The mean loss for each epoch (E) across the five folds (K) is calculated using the formula provided in Eq. 5.2. A significant decrease in training loss is observed, indicating that the model is effectively

minimizing the error on the training dataset over the epochs. Conversely, the validation loss decreases much more gradually and starts to plateau, which may suggest that while the model is learning, it's not improving as significantly on the validation set after the second epoch. This could be a sign that the model is not benefiting from further training on the same data or that it's starting to memorize the training data rather than learning to generalize.

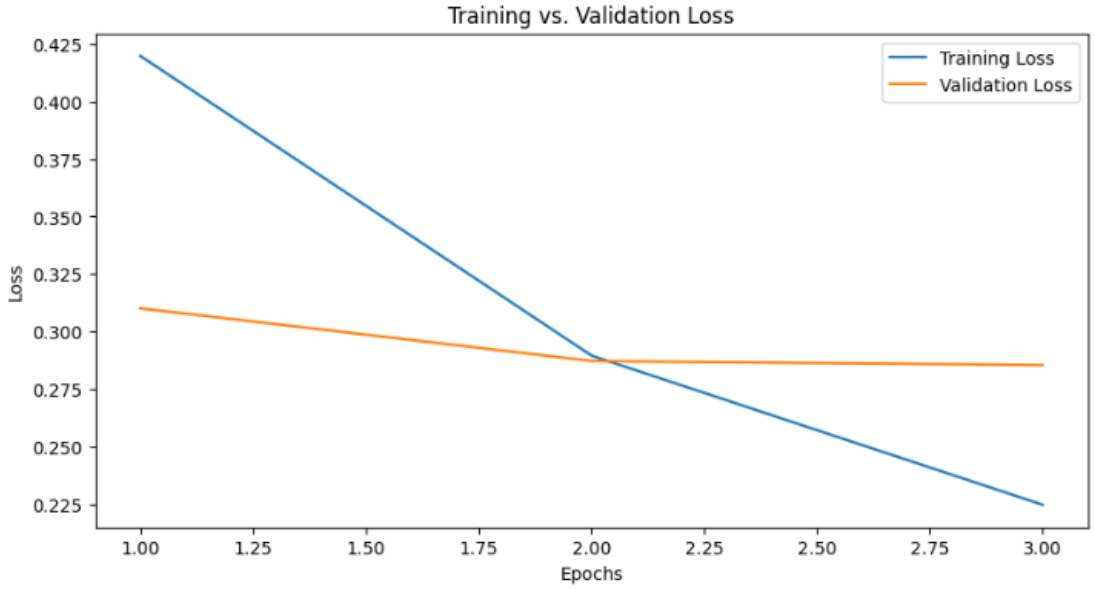

Figure 5.12: Comparison of average training and validation loss across epochs, based on five-fold cross-validation.

These graphs, displaying the mean accuracy and loss from the cross-validation, provide insights into the model's training dynamics and its challenges with generalization. To enhance the model's performance, introducing more high-quality images or implementing overfitting prevention techniques such as regularization and dropout could be beneficial. It's important to note that optimizing the detection and classification model performance is beyond this thesis's scope.

The primary goal focus on the functionality of the website and the accompanying tool. As mentioned, the tool is designed to allow users to upload images of carpets and determine their suitability for environments friendly to individuals with dementia. The website will be presented in the next section.

5.2 Web Application Interface and Functionality

The initial landing page of the web application, as shown in Fig 5.13, presents a clean interface that welcomes users upon their arrival. At the top of the page, the navigation bar provides easy access to various sections of the application, including 'Features', 'Documentation', and 'FAQ', alongside a 'Login' button, offering users a straightforward path to authenticate in order to access more personalized features, including the object detection functionality.

In general the landing page consists of a layout with header, main content and footer. The design is intentionally simplistic, avoiding call-to-action prompts or a

hero section, a decision made to reflect the educational intent of the project and to maintain the focus on the operational aspects of the tool developed rather than promotional elements.

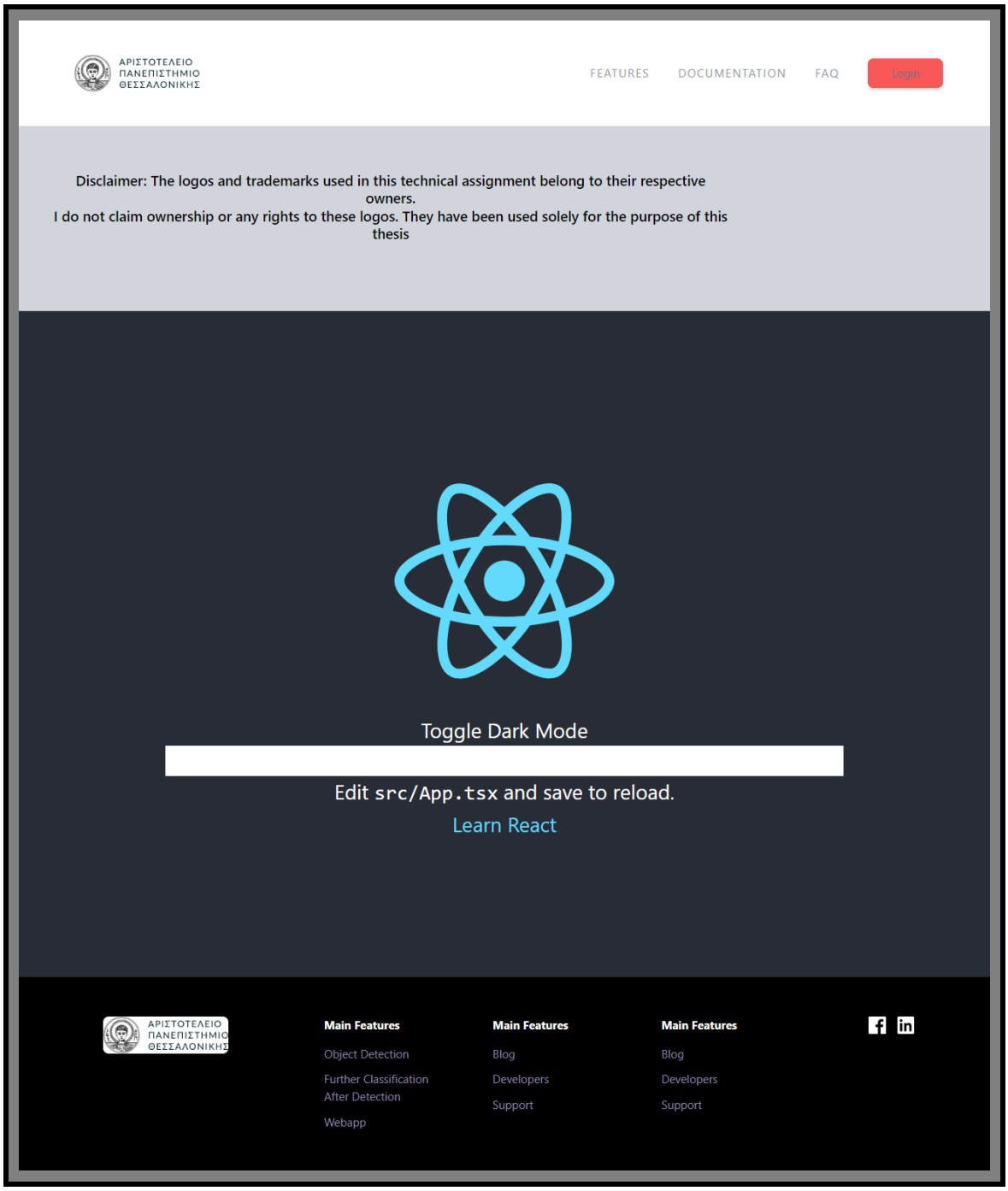

Figure 5.13: HomePage

In Section 4.1, the workflow of the web application was explained using sequence diagrams that include both front-end and back-end processes. The next section will present screenshots that correspond to these diagrams, illustrating the front-end functionality as experienced by the user.

5.2.1 Authentication in Frontend

The authentication mechanism within the web application is designed to be straightforward and it offers three key functions on the front-end: login, sign up, and logout. Users begin their interaction with the application from the homepage by clicking the "Login" button, which redirects them to the authentication pages. Here, they have the option to sign up by providing their credentials to create a new account and then login in by entering their credentials and gain access to the features of the application. For security and privacy, the application also provides a logout button. When users click logout, their session is terminated and they are redirected to the Landing Page.

As depicted in Fig. 5.14, users without an account are presented with the option to create one. However, users with existing accounts can access the application by entering their credentials and clicking the "Login" button. The login and signup form is identical.

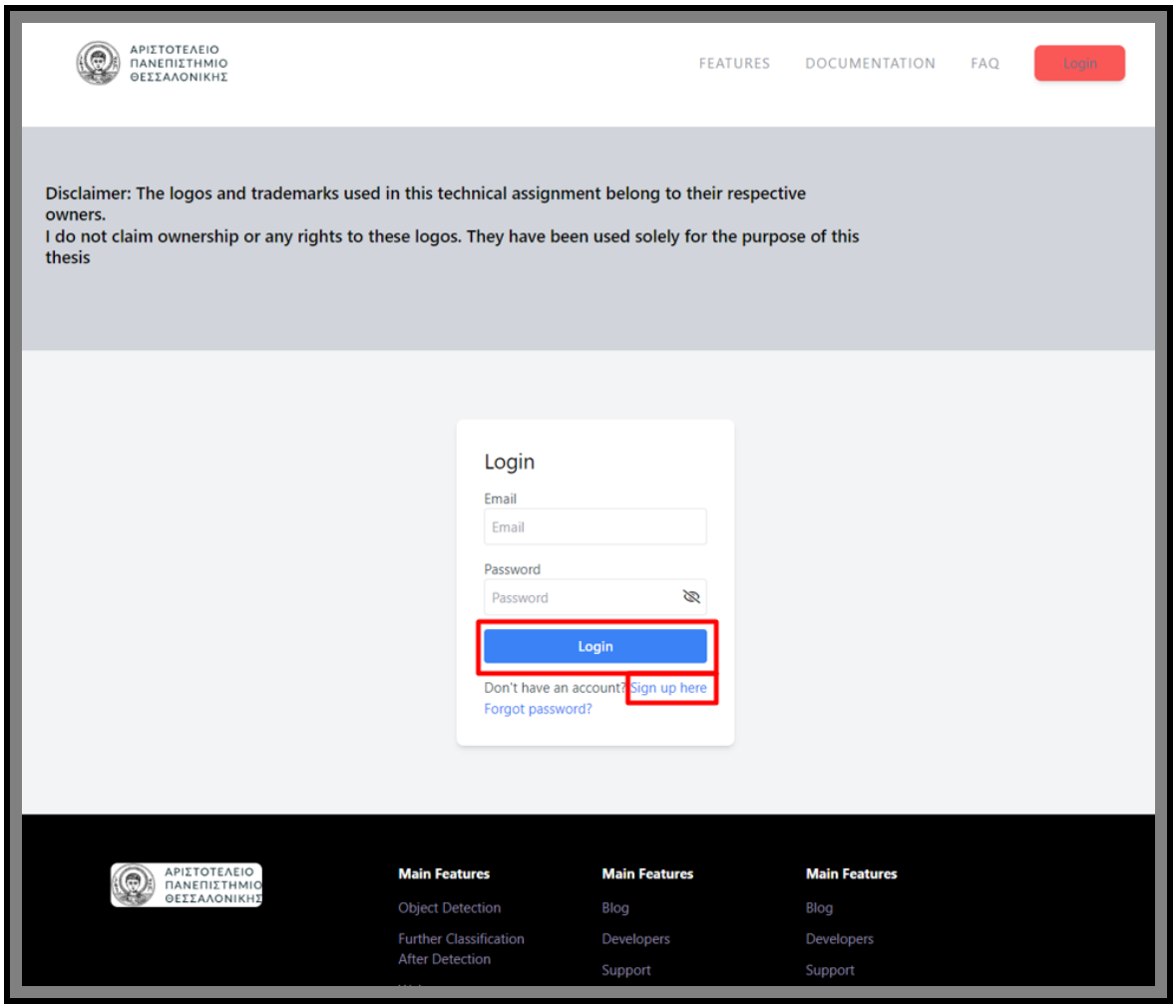

Figure 5.14: Login/Signup Page

Upon successful authentication, the user is redirected to their personalized dashboard within the application. As shown on the left side of the Fig. 5.15, the sidebar menu is responsible for navigating to the various functionalities and features. Noticeably, the 'Logout' button is accessible for users to securely exit their session. In

addition to logout, the sidebar presents other several options:

- **Account**, where users can view their personal information.
- **Settings**, allowing users to toggle between a dark or light theme.
- The core functionalities of the application are encapsulated in:
    1. **Detection**, where users can engage with the detection features
    2. **My Image Folders**, which houses results from detections and analytics

In the subsequent subsection, the 'Detection' feature will be explored with an emphasis on its operation and user interface.

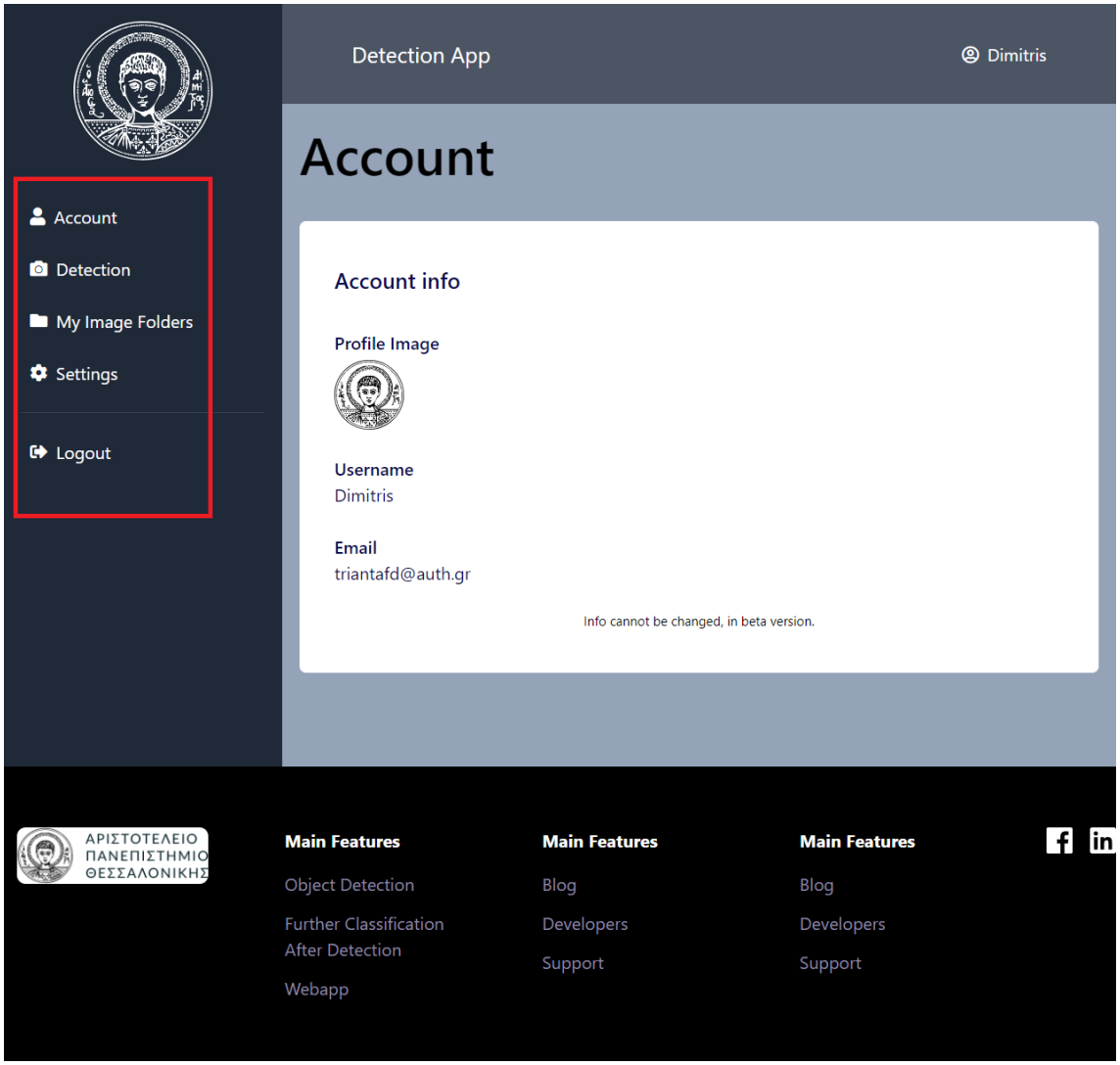

Figure 5.15: Main Application

5.2.2 Detection in Frontend and Storing Images

The 'Detection' feature of the web application comprises two advanced functionalities that will be illustrated through front-end screenshots.

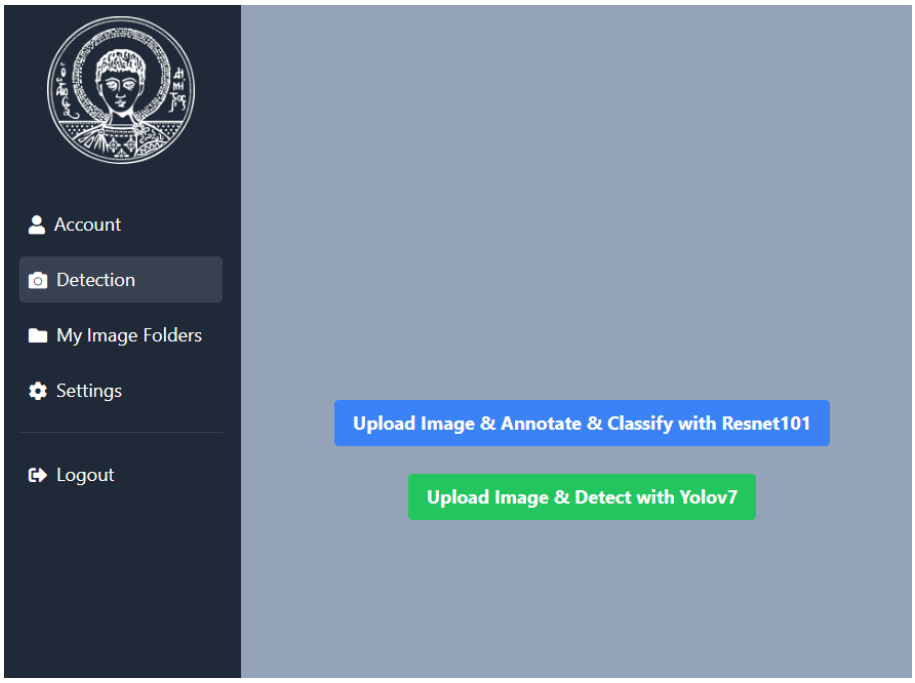

Figure 5.16: Detection Options, First option is to upload photo and detect using yoloV7. Second option is to upload photo annotate by drawing and use classification to detect

- **Object Detection with YOLOv7**: The first functionality involves using the sophisticated YOLOv7 algorithm, re-trained on a custom dataset. Users chose the option "Upload Image & Detect with Yolov7" in Fig. 5.16. As depicted in Fig. 5.18, the process is straightforward, consisting of two primary steps: firstly, users upload an image by selecting "Upload a photo" and, secondly, they initiate the detection by clicking "Detect image." The outcome of the detection, displayed on the left, showcases the algorithm's capability to recognize specific items, in this case, carpets. It is noteworthy that the algorithm is exclusively trained to identify carpets, with the label "good carpet" indicating that the identified carpet is deemed non-disruptive for individuals with dementia and is suitable for creating a dementia-friendly environment.
- **Interactive Object Labeling and Classification**: The second functionality allows users to annotate the images by drawing around objects and assigning labels and then specific classification algorithms run to give the results. Users chose Upload Image & Annotate & Classify with Resnet101 for this option in Fig. 5.16 This feature empowers users to interact directly with the image analysis process through a five-step workflow:
  1. **Upload an Image**: Users initiate the process by uploading an image using the 'Upload Image' button.
  2. **Enable Drawing**: By clicking 'Enable Drawing', users activate the tool to draw rectangles around objects within the uploaded image. This step is crucial for identifying items to determine if they are unobtrusive and can be safely integrated into dementia-friendly environments without causing confusion for individuals with dementia.

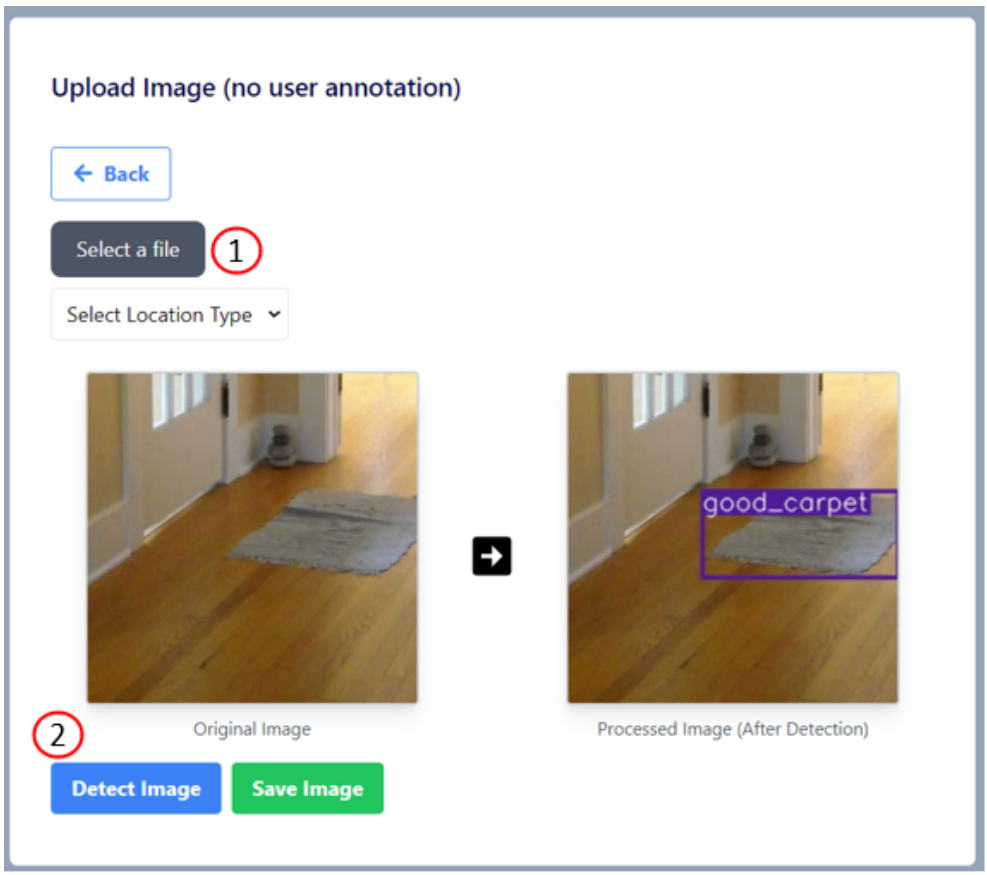

Figure 5.17: Detection using Yolov7 (no user annotation) with two simple steps. (1) Upload a photo by clicking select a file. (2) Click detect image. The image after detection appears on the left. As discussed the algorithm is only trained to recognize carpets. The label good carpet indicates that it does not confuse a person with dementia and can be used to create a dementia friendly environment

3. **Draw Rectangles**: Users draw the rectangles around the objects that they want the system to analyze.
4. **Label Objects**: After drawing the rectangles, users assign labels to these objects, specifying the class they belong to. This step is crucial for the system to ensure the appropriate classification algorithm is applied, selecting from among the existing pre-trained classification networks designed for these classes (currently only for carpets).
5. **Detect**: With the objects labeled, users can proceed to click the 'Detect' button to start the classification process.

This approach helps in evaluating whether the marked objects maintain a clear and navigable environment for individuals with dementia.

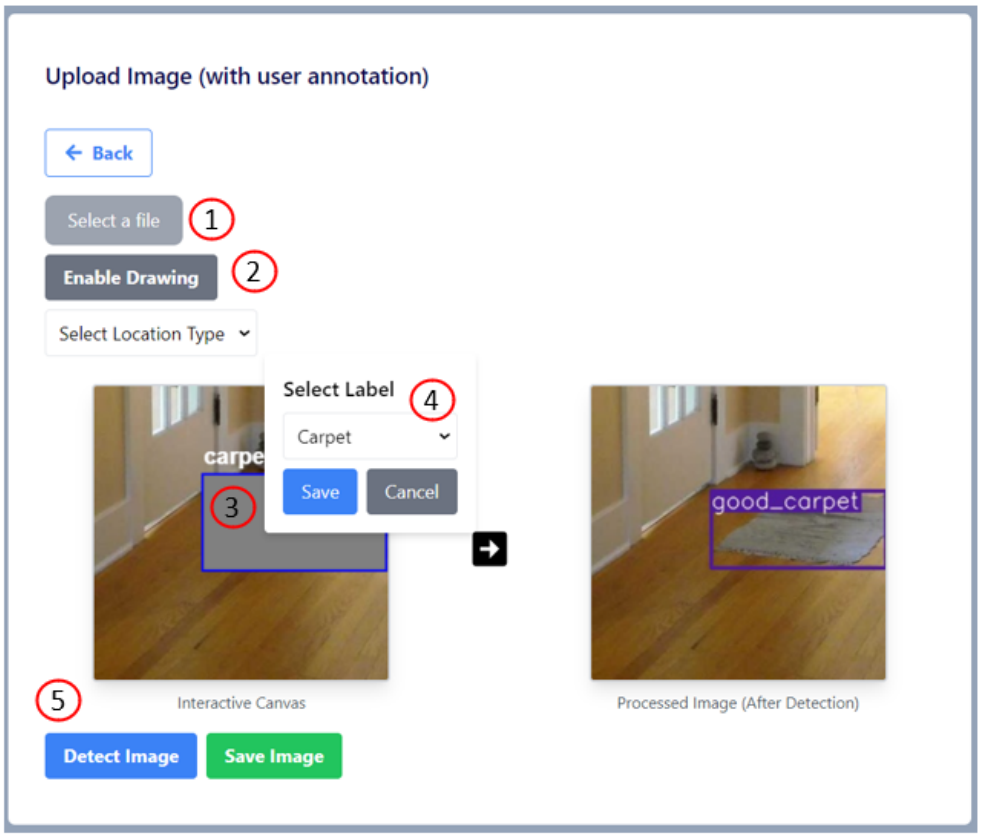

Figure 5.18: The Interactive Object Labeling and Classification workflow: (1) Image upload for analysis; (2) 'Enable Drawing' to commence marking relevant objects; (3) Outline of target objects with rectangles to assess their suitability in dementia-friendly settings; (4) Assigning labels to these objects, essential for algorithmic classification; (5) Executing the detection process with the annotated data. This approach helps in evaluating whether the marked objects maintain a clear and navigable environment for individuals with dementia.

5.2.3 User Image Storage: Viewing, Deletion, and Analytics in Frontend

The image file server within the web application serves four fundamental purposes that enhance the user experience on the front end: storing images, viewing them, deleting unwanted files, and providing analytics on the dementia-friendliness of the stored images. In the subsequent descriptions and screenshots it will be explained how these features function in the front-end environment.

Image Storage Process: The image storage process comprises of 4 steps (5.19. Initially, users select their preferred method for object detection. Upon making this selection, they upload the photo intended for detection, aiming to assess its suitability in dementia-friendly environments. Following the detection phase, users are prompted to classify the photograph by assigning a 'Location Type', such as 'house' or 'building', to specify its origin. To conclude the process, users click the 'Save' button to store the photograph in the ImageFileServer, alongside its detection annotations, metadata, and designated location type.

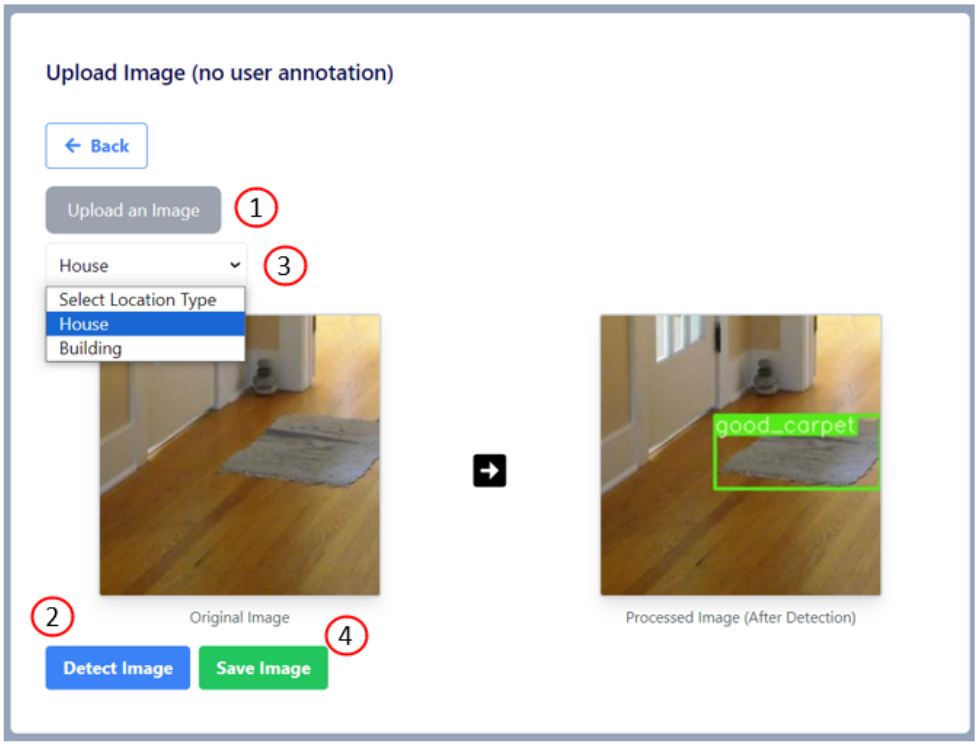

Figure 5.19: Workflow of image storage: (1) Uploading an image for analysis; (2) Detecting objects within the image; (3) Selecting a Location Type to specify the image's origin; (4) Saving the detected image along with metadata retrieved from the detection server and the designated location type provided by the user.

View Images & Analytics & Delete Images: In the user interface for viewing images, analytics, and deleting images, users can manage their uploaded content. Before accessing this page, users must select their desired image folder containing the detected photos they have saved. These folders correspond to different location types, such as 'house', 'building', or 'dementia care homes', allowing users

to navigate through their uploaded images based on their origin and assess their dementia-friendliness. (Fig. 5.20)

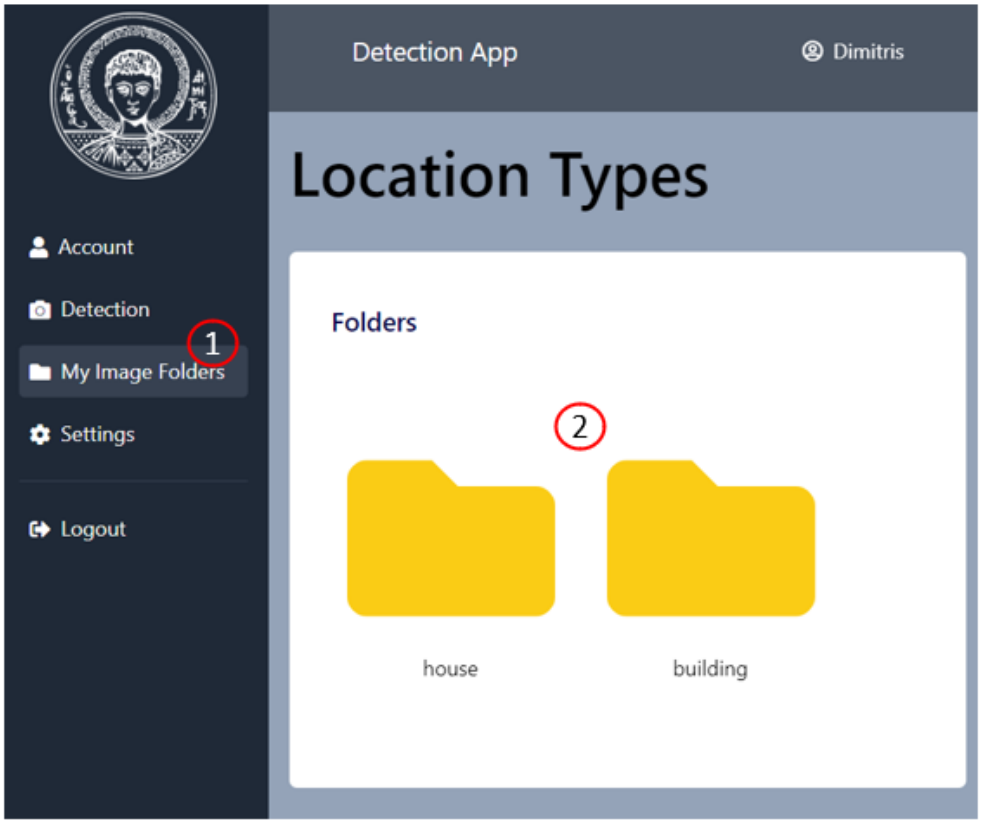

Figure 5.20: In order users to view their uploaded images the have to (1) click on 'My Image Folders' in the sidebar menu and then (2) click the desired folder the want to view.

Upon selecting a folder, users are directed to a page where they can utilize various functionalities 5.21:

1. **View Images:** Users can browse through the images within the selected folder, visually inspecting each photo and its details.
2. **Analytics:** Users have the option to delve into analytics and view statistics derived from the uploaded images. For instance, users can review the distribution of images per class (e.g., images containing carpets), distinguish between 'good' and 'bad' images within each class, and ascertain the percentage of images deemed dementia-friendly.
3. **Delete Images:** Users can manage their image repository by selecting and deleting images directly from the interface. This functionality is accessible via a bin icon that is positioned on the right-hand side of the screenshot 5.21

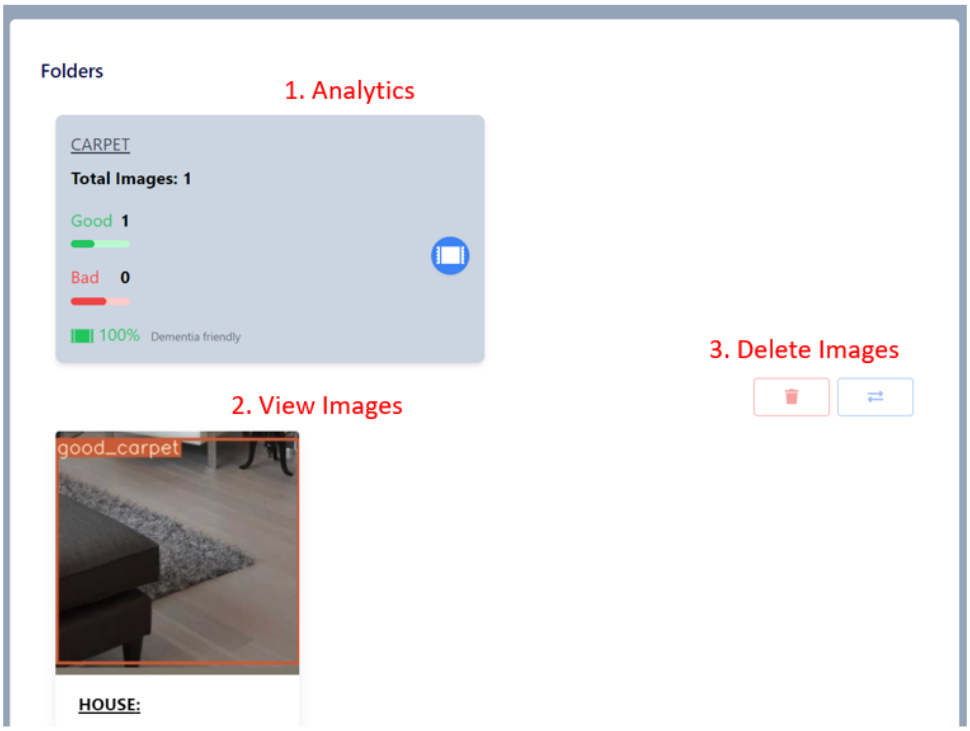

Figure 5.21: The user interface where users can (1) View Images, (2) ViewAnalytics and (3) Delete unwanted photos

6

Conclusions and Future Work

6.1 Summary of Findings

This thesis presents the development of a web-based tool designed to aid in the creation of dementia-friendly environments. With the rising number of individuals diagnosed with dementia, it is imperative to adapt public spaces to their needs. Specific objects in public spaces, such as complex flooring, like carpets with patterns, can disorient people with dementia and affect their navigation and psychological well-being.

Currently, the thesis concentrates on analyzing carpets to identify potentially confusing patterns and determine their suitability for environments occupied by people with dementia. To facilitate this assessment, detection and classification models were utilized to automate the evaluation process for these floor coverings.

The detection model, which was based on YOLOv7, faced some challenges, possibly due to dataset limitations. Despite dataset refinements, the model's ability to accurately detect objects was not perfect. The classification task, implemented using a ResNet-101 architecture, proved to be more straightforward. It bypassed the need for object detection and directly classified the images as dementia-friendly or not. After further refinement and reshuffling of the dataset, the classification model demonstrated much-improved results.

It should be noted that the the tool's functionality is very straightforward as allows users to upload images and specify the type of location (e.g., home, care facility, hospital). The tool then applies the detection model to identify objects within the image and assesses their suitability for individuals with dementia. If no objects are detected, users have the option to manually outline objects and label them, running the classification model to determine their appropriateness. Users have the convenience of saving their findings in folders organized by location type and can review these alongside analytics that reflect the percentage of dementia-friendly

objects in the images. Apparently, there is still room for improvement in various directions that will be presented in the next section.

6.2 Future Directions

6.2.1 User Interface and Experience Improvement

The current state of the website indicates that there's room for improvement in both design and accessibility. To enhance user engagement and ensure accessibility, a significant redesign of the application is crucial. This will focus on creating an interface that is both intuitive for users and compliant with accessibility standards.

At the moment, the process for object detection and determining if an item is suitable for individuals with dementia can be cumbersome. Users upload an image, and if the object isn't recognized, they must switch to a different page to draw and label the object themselves. This process should be streamlined and that can be achieved by integrating both functionalities on a single page in order to offer a smoother user experience that keeps users engaged and reduce frustration.

Lastly, our tool allows users to upload and analyze only one photo at a time for dementia-friendliness based on the objects present. This feature should be enhanced to enable batch uploading, where users can submit multiple photos simultaneously for analysis. This will significantly improve efficiency and user satisfaction.

6.2.2 Expansion of Object Categories

The tool can be further developed to include a wider variety of objects that significantly impact the dementia-friendly aspect of indoor spaces. Currently, it mainly focuses on flooring evaluating carpets, but future improvements will include adding objects such as mirrors and doors. By expanding the range of objects assessed, the tool can offer a more thorough evaluation of indoor environments, considering factors that affect the behavior and psychology of individuals with dementia. Additionally, enhancing image quality is crucial for accurate assessments and lastly the upgrade of the image database with better-quality images will improve the tool's effectiveness in analyzing and assessing indoor spaces for dementia-friendliness.

6.2.3 Exploring Alternative Training and Evaluation Techniques

While deep learning models offer powerful solutions for pattern recognition and classification, exploring alternative methodologies could possibly yield beneficial results, especially in contexts with dataset limitations or specific computational constraints. One such approach involves leveraging bare OpenCV [45] techniques for feature extraction and analysis to infer a carpet's surface complexity, determining its dementia-friendliness without having to rely on deep learning frameworks. By implementing algorithms focused on texture and pattern complexity, it is possible to assess the dementia-friendliness of carpets and similar objects with potentially greater efficiency and lower resource requirements.

6.2.4 Deployment

In this thesis, a modular architecture that deconstructs the overall system into discrete, independently functioning components was developed. This architecture comprises several key applications, including:

1. **A client application built with React.js,**
2. **An authentication service,**
3. **An image file server application, and**
4. **An image detection service.**

To manage these components effectively, a Docker Compose script was employed for container orchestration. While the client application was initially run with npm start for development, transitioning to a production-grade Nginx server would enhance stability. Additionally, setting up a local Docker image registry would streamline image management. The current thesis offers a strong starting point for future developments, especially in microservices architecture. Adopting Kubernetes as an orchestration tool is the next logical step to improve scalability and manageability.

In summary, while the current setup supports the system's core functions well, strategic enhancements such as transitioning to a production-grade server and implementing a local Docker image registry can improve scalability and reliability. Lastly, exploring the adoption of Kubernetes as an orchestration tool presents an opportunity to effectively manage growth and fulfill future user requirements.

Bibliography

[1] B. de Boer, B. Bozdemir, J. Jansen, M. Hermans, J. P. Hamers, and H. Verbeek, "The homestead: developing a conceptual framework through co-creation for innovating long-term dementia care environments," *International Journal of Environmental Research and Public Health*, vol. 18, no. 1, p. 57, 2021.

[2] J. M. Wiener and F. Pazzaglia, "Ageing-and dementia-friendly design: theory and evidence from cognitive psychology, neuropsychology and environmental psychology can contribute to design guidelines that minimise spatial disorientation," *Cognitive processing*, vol. 22, no. 4, pp. 715–730, 2021.

[3] World Health Organization, "World health organization website," https://www.who.int/, 2024, [Online; accessed 18-January-2024].

[4] K. Day, D. Carreon, and C. Stump, "The therapeutic design of environments for people with dementia: a review of the empirical research," *The gerontologist*, vol. 40, no. 4, pp. 397–416, 2000.

[5] F. Kelly, A. Innes, and O. Dincarslan, "Improving care home design for people with dementia," *Journal of Care Services Management*, vol. 5, no. 3, pp. 147–155, 2011.

[6] M.-y. Leung, C. Wang, and I. O. Famakin, "Integrated model for indoor built environment and cognitive functional ability of older residents with dementia in care and attention homes," *Building and Environment*, vol. 195, p. 107734, 2021.

[7] G. Marquardt, "Wayfinding for people with dementia: a review of the role of architectural design," *HERD: Health Environments Research & Design Journal*, vol. 4, no. 2, pp. 75–90, 2011.

[8] M.-y. Leung, C. Wang, and I. Y. Chan, "A qualitative and quantitative investigation of effects of indoor built environment for people with dementia in care and attention homes," *Building and Environment*, vol. 157, pp. 89–100, 2019.

[9] F. Ferdous, "Positive social interaction by spatial design: a systematic review of empirical literature in memory care facilities for people experiencing dementia," *Journal of aging and health*, vol. 32, no. 9, pp. 949–961, 2020.

[10] P. Barrett, M. Sharma, and J. Zeisel, "Optimal spaces for those living with dementia: Principles and evidence," *Building Research & Information*, vol. 47, no. 6, pp. 734–746, 2019.

[11] G. Cipriani, S. Danti, L. Picchi, A. Nuti, and M. D. Fiorino, "Daily functioning and dementia," *Dementia & neuropsychologia*, vol. 14, pp. 93–102, 2020.

[12] J. Joyce, J. Ryan, A. Owen, J. Hu, J. McHugh Power, R. Shah, R. Woods, E. Storey, C. Britt, R. Freak-Poli *et al.*, "Social isolation, social support, and loneliness and their relationship with cognitive health and dementia," *International Journal of Geriatric Psychiatry*, vol. 37, no. 1, 2022.

[13] National Institute of Neurological Disorders and Stroke, "Dementia," 2024, [Online; accessed 18-January-2024]. [Online]. Available: https://www.ninds.nih.gov/

[14] C. Wiegmann, I. Mick, E. J. Brandl, A. Heinz, and S. Gutwinski, "Alcohol and dementia–what is the link? a systematic review," *Neuropsychiatric Disease and Treatment*, pp. 87–99, 2020.

[15] C. Quinn, J. A. Pickett, R. Litherland, R. G. Morris, A. Martyr, L. Clare *et al.*, "Living well with dementia: What is possible and how to promote it," *International Journal of Geriatric Psychiatry*, vol. 37, no. 1, 2022.

[16] Alzheimers Society in UK, "Dementia," 2024, [Online; accessed 18-January-2024]. [Online]. Available: https://www.alzheimers.org.uk/

[17] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[18] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5-6, pp. 183–197, 1991.

[19] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," *Advances in neural information processing systems*, vol. 2, 1989.

[20] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[21] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[22] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for neural networks for image processing," *arXiv preprint arXiv:1511.08861*, 2015.

[23] Y. Sai, R. Jinxia, and L. Zhongxia, "Learning of neural networks based on weighted mean squares error function," in *2009 Second International Symposium on Computational Intelligence and Design*, vol. 1. IEEE, 2009, pp. 241–244.

[24] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE access*, vol. 8, pp. 4806–4813, 2019.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[26] L. Prechelt, "Early stopping-but when?" in *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

[27] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.

[28] J. Bouwman, "Quality of regularization methods," *DEOS Report 98.2*, 1998.

[29] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[30] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[31] M. Awad, R. Khanna, M. Awad, and R. Khanna, "Support vector machines for classification," *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, pp. 39–66, 2015.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[33] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[34] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[35] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[36] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[37] G. Jocher, L. Changyu, A. Hogan, L. Yu, P. Rai, T. Sullivan *et al.*, "ultralytics/yolov5: Initial release," *Zenodo*, 2020.

[38] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[39] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] Stephen Grider, "Stephengrider github repository," https://github.com/StephenGrider, 2024, accessed on February 12, 2024.

[42] "Unavailable image," https://images.cv, accessed: insert access date here.

[43] B. Setiyono, D. A. Amini, and D. R. Sulistyaningrum, “Number plate recognition on vehicle using yolo-darknet,” in *Journal of Physics: Conference Series*, vol. 1821, no. 1. IOP Publishing, 2021, p. 012049.

[44] chinakook, “labelimg2,” Accessed: 2024-01-25, gitHub repository. [Online]. Available: https://github.com/chinakook/labelImg2

[45] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O’Reilly Media, Inc.", 2008.