Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Bachelor Thesis

Detection of user interface elements on a webpage

Polina Medynska

© 2023 CZU Prague

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

BACHELOR THESIS ASSIGNMENT

Polina Medynska

Informatics

Thesis title

Detection of user interface elements on a webpage

Objectives of thesis

The main objective of this work is to detect elements of the user interface on the image of the webpage, e.g., text, graphic elements, or form items.
The partial objectives:
- To create a literature review with a focus on computer vision.
- To find public or create and annotate an experimental data set.
- To select model/s for experimental evaluation.

Methodology

The methodology of solving the theoretical part of the Bachelor thesis will be based on the study materials and analysis of professional information sources. Firstly, the experimental data set will be gathered either by using publicly available data or by creating and annotating a new set. Based on the theoretical part, the most appropriate method/s for the implementation will be selected. After that, the method/s will be evaluated using chosen metrics. Based on the synthesis of theoretical knowledge and the results of the practical part, conclusions will be formulated.

The proposed extent of the thesis

40 – 50 pages

Keywords

Computer vision, user interface, deep learning, OCR, recognition

---

Recommended information sources

Himanshu, S. Allahabad, U. Pradesh. Practical Machine Learning and Image Processing. 2019. ISBN: 978-1-4842-4149-3.

Jiang; Xiaoyue; Hadid; Adbenour. Deep learning in object detection and recognition. Limited 2019. ISBN: 9789811051517.

Yannis M., Barbara H., Lazaros I., Ilias M. Artificial Neural Networks and Machine Learning ICANN 2018. ISBN: 978-3-030-014118-6.

Expected date of thesis defence

2022/23 SS – FEM

The Bachelor Thesis Supervisor

Ing. Jan Masner, Ph.D.

Supervising department

Department of Information Technologies

Electronic approval: 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Head of department

Electronic approval: 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 08. 10. 2023

Declaration

I declare that I have worked on my bachelor thesis titled " Detection of user interface elements on a webpage" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the bachelor thesis, I declare that the thesis does not break any copyrights.

In Prague on 29/11/2023 _____

Acknowledgment

I would like to thank Jan Masner for supervision during my work on the thesis and also thank my brother for his advice and great support throughout my studies.

Detection of user interface elements on a webpage

Abstract

This thesis explores the application of computer vision for the detection of user interface elements on webpages. It delves into foundational concepts of artificial intelligence, machine learning, neural networks, deep learning, and computer vision. The methodology of the work involves the preparation of an experimental dataset, forming the basis for evaluating YOLO object detection models.

The literature review spans these key domains, establishing a robust foundation for investigating element detection on images of web pages. In the practical part, models were trained and evaluated using publicly available data. The results section includes an analysis of the used approach, a comparison of models, and suggestions for further development.

Keywords: Computer vision, user interface, detection, recognition, artificial intelligence, neural networks.

Detekce prvků uživatelského rozhraní na webové stránce

Abstrakt

Tato práce zkoumá aplikaci počítačového vidění pro detekci prvků uživatelského rozhraní na webových stránkách. Ponoří se do základních konceptů umělé inteligence, strojového učení, neuronových sítí, hlubokého učení a počítačového vidění. Metodika práce zahrnuje přípravu experimentální datové sady, která tvoří základ pro hodnocení modelů detekce objektů YOLO.

Přehled literatury zahrnuje tyto klíčové domény a vytváří robustní základ pro vyšetřování detekce prvků na obrázcích webových stránek.V praktické části byly modely proškoleny a vyhodnoceny pomocí veřejně dostupných dat. Sekce výsledky obsahuje analýzu použitého přístupu, porovnání modelů a návrhy na další vývoj.

Klíčová slova: Počítačové vidění, uživatelské rozhraní, detekce, rozpoznávání, umělá inteligence, neuronové sítě.

Table of content

1 Introduction

As technology evolves and becomes more sophisticated, it requires more complex tools to be developed. There are multiple approaches to solving this problem, and one of the most promising is artificial intelligence. It is often undertaken in conjunction with machine learning and data analytics [1]. Artificial intelligence algorithms are designed to make decisions in a way similar to humans. An exponential growth of information and data in the world leads to an opportunity to use it for developing such algorithms. Machine learning is a field of science that specifically focuses on gathering, processing, and analysing datasets and using them for training machines to select and perform actions based on a certain input. Because of the availability of diverse data sources, different types of data now exist for training, such as images, videos, and numerical and textual datasets. Deriving information from visual inputs is a challenge that computer vision as a scientific field deals with. It focuses on developing methods to acquire, process, and analyse such inputs as well as extract high-dimensional data. Computer vision has multiple subdomains, which include object detection, object recognition, event detection, video tracking, etc.

Specifically in the software domain, object detection and recognition of images are widely used. As most of the software is created so that human beings can interact with it, one of the main requirements is its intuitive, responsive, and effective interface. Modern user interfaces are mostly visual and provide users with extensive functionality. Another important characteristic is their adaptability to the current needs of the user. It became possible due to the analysis of human-computer interaction, which means the interaction of the user with the interface. Graphical user interfaces (GUI) evolved from rigid interfaces with images, buttons, and texts to ones that dynamically adapt to the current tasks, adjust input and output processes, and allow for complex interactions in a user-friendly manner. The process of interaction between the user and the interface itself contains a lot of information, that can be derived through computer vision and machine learning. It can be used immediately to predict future user actions and adjust accordingly, or, saved and analysed later for further improvement of the UI.

Thus, the detection and recognition of user interface elements is the foundation of many software engineering tasks.

In this work, different recognition methods are analysed and compared.

2 Objectives and Methodology

2.1 Objectives

The main objective of this work is to detect elements of the user interface on the image of the webpage, e.g., text, graphic elements, or form items.

The partial objectives:

- To create a literature review with a focus on computer vision.
- To find the public or create and annotate an experimental data set.
- To select model/s for experimental evaluation

2.2 Methodology

The methodology of solving the theoretical part of the Bachelor thesis is based on the study materials and analysis of professional information sources. Firstly, the experimental data set is gathered either by using publicly available data or by creating and annotating a new set. Based on the theoretical part, the most appropriate method/s for the implementation is selected. After that, the method/s is evaluated using chosen metrics. Based on the synthesis of theoretical knowledge and the results of the practical part, conclusions are formulated.

3 Literature Review

3.1 Artificial intelligence

3.1.1 The concept of artificial intelligence

Artificial intelligence is a multidisciplinary field of computer science that focuses on creating intelligent machines capable of imitating human behaviour to solve specific tasks. It involves training these machines using received information and is closely connected to the study of the properties of the human brain. Researchers believe that understanding the principles of the brain is crucial in achieving the creation of artificial intelligence. Within the field of artificial intelligence, researchers employ a diverse range of techniques and methodologies to simulate various cognitive processes observed in humans, such as learning, reasoning, problem-solving, perception, and decision-making. By simulating these processes, we aim to develop machines that possess human-like abilities in learning, thinking, and decision-making. Through the emulation of the cognitive processes occurring in the human brain, we can create intelligent machines capable of replicating these abilities. The main components that make up the bulk of artificial intelligence systems are natural language processing; knowledge representation; machine learning tools; computer vision; automated reasoning tools; and robotics tools. Artificial intelligence and machine learning are often confused with each other, but they are two different things. Machine learning is a branch of artificial intelligence that focuses on developing algorithms that can learn from data and improve their performance over time without being programmed. Some artificial intelligence systems use machine learning to achieve their goals, but others do not. [1]

3.1.2 Key components of artificial intelligence

Two pivotal components that play a crucial role in advanced artificial intelligence are machine learning and computer vision. Machine learning, a branch of artificial intelligence, creates algorithms that allow machines to learn from data, improving performance autonomously. Through data analysis, these algorithms make predictions or decisions without explicit programming. This is vital for artificial intelligence's experiential learning and task enhancement. Computer vision, a branch of artificial intelligence, empowers machines to interpret visual data. It develops algorithms for extracting information from images/videos, understanding content, and comprehending the

visual environment. Tasks include object recognition, image classification, detection, and segmentation. Machine learning, especially deep learning, significantly advances computer vision. Deep learning models like Convolutional Neural Networks excel in image tasks. Computer vision also aids machine learning by providing complex datasets for training. In GUI (graphical user interface) detection research, computer vision identifies graphical elements (buttons, menus) in user interfaces. Coupled with machine learning, it accurately detects and localizes GUI components. Trained on GUI datasets, models offer insights for analysis and automation. [1]

3.2 Machine learning

3.2.1 Fundamental concepts of machine learning

Machine learning is a common application of artificial intelligence in modern businesses and is expected to be a crucial component of the IT (information technology) strategy for many enterprises. It plays a significant role in transforming the IT industry, applicable to various workflows in software development, research, production processes, and even the products themselves. [2]

Machine learning is based on three equally important components:

1. Data, collected through various means. The efficiency of machine learning and the accuracy of future results increase with a larger volume of data.
2. Features, defining the parameters on which machine learning operates.
3. Algorithm, serving as the foundation of machine learning. These are sets of rules and procedures that handle the data, producing models capable of predictions or classifications. The selection of a machine learning method impacts the accuracy, speed, and size of the final model.

In terms of types of learning, there are four categories:

1. Supervised Learning:

The machine learning algorithm learns from input-output pairs supervised learning algorithms experience a dataset containing features, and each example is also associated with a label or target. The algorithm learns to map inputs to outputs by finding patterns and relationships in the data. The goal is to train a model that can accurately predict outputs for new, unseen inputs. Example: suppose there is a dataset of customer reviews for a product, where each review is accompanied by a sentiment label indicating whether the review is positive or negative. A supervised learning algorithm can analyse this dataset and learn to

classify future customer reviews as either positive or negative based on the text content. This can be particularly useful for sentiment analysis in customer feedback analysis or product review platforms, helping businesses understand the overall sentiment of their customers towards their products or services. [3]

2. Unsupervised Learning.

It involves learning patterns and structures in unlabelled data without explicit feedback or labels. The algorithm explores the data and discovers hidden patterns, relationships, or clusters within it. Clustering is a common unsupervised learning task where similar examples are grouped based on their intrinsic similarities.

3. Semi-supervised learning.

Is a type of machine learning that combines elements of supervised and unsupervised learning. It leverages both labelled and unlabelled data to train a model. While supervised learning relies solely on labelled data and unsupervised learning works with unlabelled data, semi-supervised learning takes advantage of the additional information provided by unlabelled examples to improve model performance. The idea behind semi-supervised learning is that the unlabelled data can help in discovering and modelling the data's underlying distribution. By incorporating the unlabelled examples, the model can better generalize and make more accurate predictions on new, unseen data.

4. Reinforcement Learning.

It is a learning process where an agent explores and interacts with its environment, receiving positive or negative feedback based on its choices, and continuously adapts its behaviour to maximize its overall rewards. Through trial and error, the agent learns to take actions that lead to higher rewards and avoid actions that result in penalties. Reinforcement learning is often used in scenarios where an agent must make sequential decisions, such as game playing or robot control.

The most common supervised learning tasks are:

1. Regression is a machine learning approach that utilizes data to predict continuous numerical values by modeling the correlation between an outcome variable and its contributing variables. The goal is to create a mathematical model that can accurately predict the value of the dependent variable based on the given independent variables. Regression models can help understand the relationship between variables and make predictions about future outcomes.

2. Classification, on the other hand, is a machine-learning technique used for predicting discrete categories or labels. It involves training a model with a set of labelled examples, where each example is associated with a specific class or category. The model then learns to classify new, unseen examples into one of the predefined categories. Classification models are widely used in various applications, such as email spam filtering, image recognition, and sentiment analysis. [1]

3.3 Neural networks

3.3.1 Neural networks architecture

A neural network is a structure consisting of interconnected neurons, inspired by the biological model. This architecture enables machines to analyse and even store various information, thereby acquiring the ability to perform complex tasks. Neural networks excel at tasks that involve analytical calculations similar to those performed by the human brain. Some common applications of neural networks include:

1. Classification: Neural networks can classify data based on specific parameters. For instance, given a group of individuals, a neural network can assess factors such as age, solvency, and credit history to determine who should be granted a loan and who should not.
2. Prediction: Neural networks possess the capability to predict future outcomes. For instance, they can analyse the stock market situation to forecast whether stock prices will rise or fall.
3. Recognition: Neural networks are extensively used in recognition tasks. For example, they are employed by Google for image searches, as well as in phone cameras for facial recognition and other related functions.

In summary, neural networks emulate the biological structure of interconnected neurons, enabling machines to perform tasks involving analysis, prediction, and recognition. [3]

3.3.2 Neuron

Neurons are the fundamental units of a neural network, inspired by their biological counterparts in the human brain. They receive input signals, perform computations, and generate output signals. In a neural network, neurons are organized into layers:

- Input Layer: The input layer receives the initial data or input features and passes them to the next layer.
- Hidden Layers: Hidden layers are intermediate layers between the input and output layers. They perform computations and extract features from the input data.
- Output Layer: The output layer produces the final output of the neural network. The number of neurons in the output layer depends on the type of task the network is designed for (e.g., classification, regression).

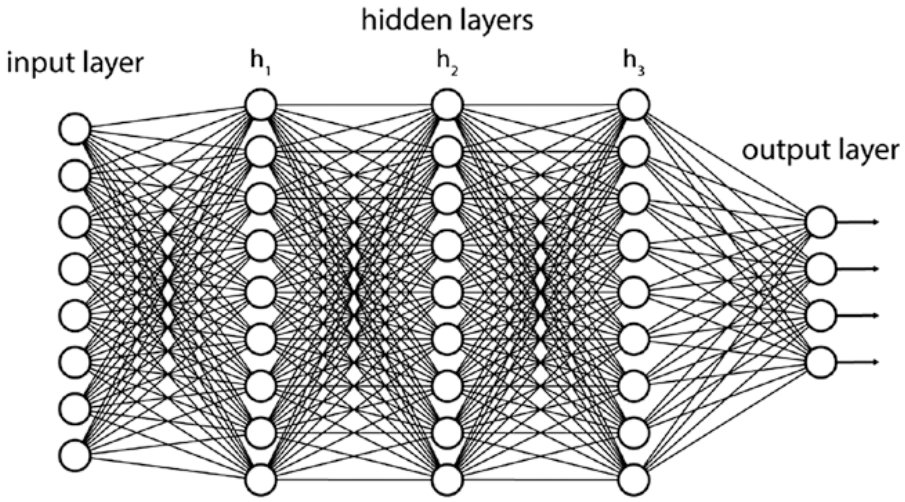


Figure 1. Deep neural networks. Source: [4]

The main computational element (model neuron) is often called a node or unit. It receives input data from some other devices or possibly from an external source. A model neuron consists of three main components:

- Inputs: A model neuron receives input signals from other neurons or external sources. Each input signal has a corresponding weight.
- Weighting: Each input signal is multiplied by its corresponding weight. Weights represent the strength of the connection between the input neuron and the model neuron.
- Activation Function: The weighted inputs are summed together, and the result is passed through an activation function. The activation function determines whether the model neuron should fire or not.

There are various types of activation functions used in model neurons, each with its characteristics. Some commonly used activation functions include:

1. Sigmoid function: Outputs a value between 0 and 1, representing the probability of the neuron firing
2. Tanh function: Outputs a value between -1 and 1, similar to the sigmoid function.

ReLU (Rectified Linear Unit): Outputs the input directly if it is positive, and outputs 0 if it is negative.

The weights of the connections between neurons in a neural network are adjusted during the training process. This adjustment is based on the error between the network's output and the desired output. The process of adjusting weights is called learning.

Neural networks are used in a wide range of applications, such as:

1. Image classification: identifying objects in images.
2. Speech recognition: converting spoken language into text.
3. Natural language processing: understanding and generating human language.
4. Machine translation: translating text from one language to another.

This concept of weighted inputs and activation functions is fundamental in neural network architecture. The single-layer perceptron (SLP) model, illustrated in Figure 2., which is the simplest form of a neural network, employs this mechanism. In SLP, the values in the input layer are multiplied by weights and a bias is added to the cumulative sum. The resulting sum is then passed through an activation function to determine the output. The SLP model is often used in classification problems, where data observations are labelled based on the inputs. It serves as a foundational concept for more advanced neural network models developed in the field of deep learning. It's worth noting that the concept of model neurons, weighted inputs, and activation functions has been studied since the 1940s by researchers such as McCulloch and Pitts. This demonstrates the longstanding history and importance of these concepts in the field of neural networks. [4]

The main computational element, commonly referred to as a node or unit, is the model neuron. It receives input data from some other devices or possibly from an external source. The unit calculates some function f of the weighted sum of its inputs, equation (1):

$$yi = f\left(\sum_{i} wij\, yij\right) \tag{1}$$

- The weighted sum $\sum_i wij\, yij$ is called the net input to unit *i*, often written *neti.*
- *Wij* refers to the weight from unit *j* to unit *i.*
- Function *f* - is the device activation function.

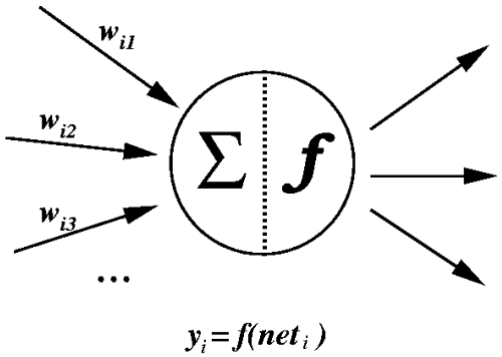


Figure 2. Single-layer perceptron network. Source: [4].

In summary, a neural network consists of interconnected neurons organized in layers. Neurons receive inputs, perform computations using activation functions, and produce outputs. Connections between neurons are established through synapses with weight coefficients. Activation functions introduce non-linearity and determine the flow of signals through the network, enabling it to learn and make predictions.

3.3.3 The work of neural networks

The input layer of neurons receives some information, which goes to the next layer through synapses. At the same time, each synapse has its coefficient weight, and any subsequent neuron in a new layer can have several inputs. Information is transmitted further until it reaches the final exit. For example, a handwriting recognition algorithm should be able to cope with a huge variety of ways of presenting data. Each digit from 0 to 9 can be written in many ways: the size and exact shape of each character can vary greatly depending on who writes and under what circumstances. The input layer is given values representing the pixels that make up the image of the handwritten digit. The output layer, in turn, predicts which symbol is depicted in the picture in Figure 3.

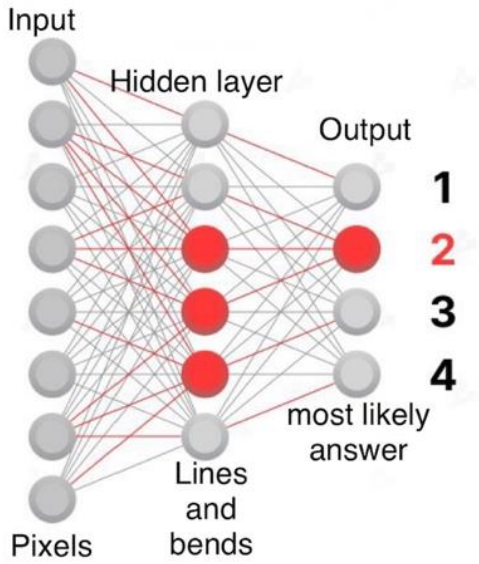


Figure 3. Layers of neural network. Source: [4]

The circles in the diagram are neurons that are organized into interconnected vertical layers. The colours of references also differ: they indicate the importance of connections between neurons. Red links increase the value when switching between layers, which increases the chance of activating the neuron to which the value enters. In the diagram, the activated neurons are shaded in red. In Hidden Layer 1, they mean that the handwritten figure image contains a certain combination of pixels resembling a horizontal line at the top of the handwritten number 3 or 7. "Hidden Layer 1" can detect feature lines and curves that make up handwritten shapes.

3.3.4 Types of neural networks

In total, there are about 30 different types of neural networks that are suitable for different types of tasks. For example, convolutional neural networks (CNNs) are commonly used for computer vision tasks, while recurrent neural networks (RNNs) are used for language processing. Each has its characteristics.

CNNs shown in Figure 4. are designed to handle the spatial structure of images effectively. They employ specific layers and operations that make them well-suited for visual data analysis. Some key components and concepts related to CNNs include:

1) Convolutional Layers: These layers consist of filters or kernels that slide across the input image, performing element-wise multiplications and summations. This process captures local patterns and features, preserving the spatial relationship between pixels.

2) Pooling Layers: Pooling layers downsample the spatial dimensions of the input, reducing its size and extracting the most important information.
3) Activation Functions: Activation functions introduce non-linearity into the network, enabling it to learn complex relationships. Popular activation functions in CNNs include the Rectified Linear Unit and its variants.
4) Fully Connected Layers: These layers connect every neuron in one layer to every neuron in the next layer, enabling higher-level feature learning and classification.

Overall, CNNs have had significant success in various computer vision tasks, achieving state-of-the-art performance on image classification challenges. They have become a fundamental tool in deep learning for visual data analysis. [11]

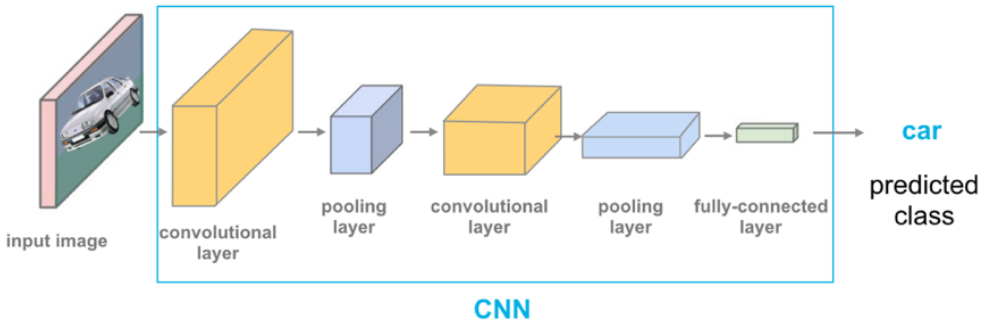


Figure 4. CNN. Source: [6]

Recurrent Neural Networks (RNNs) are a type of neural network commonly used for sequential data analysis, such as natural language processing and speech recognition. Unlike feedforward neural networks, which process data in a strictly sequential manner, RNNs have a feedback mechanism that allows information to persist and be shared across different time steps. The key feature of RNNs is their ability to capture sequential dependencies by maintaining an internal memory state or "hidden state." This hidden state serves as a memory that retains information about the previous inputs it has encountered. It allows the network to consider the context and history of the input sequence when making predictions or decisions. RNNs operate recurrently by processing one input at a time while updating the hidden state. The hidden state is updated based on the current input and the previous hidden state, combining information from both. This recurrent process enables the network to model temporal dynamics and capture long-term dependencies in the data. One common variant of RNNs is the Long Short-Term Memory (LSTM) network. LSTM networks address the vanishing gradient problem that can occur in traditional RNNs,

allowing them to effectively capture and propagate information over long sequences. LSTM networks achieve this by introducing specialized memory cells and gating mechanisms that regulate the flow of information. Another variant is the Gated Recurrent Unit (GRU), which is a simplified version of the LSTM. GRUs also incorporate gating mechanisms but with fewer parameters, making them computationally less expensive while still being effective for capturing temporal dependencies. RNNs are widely used in various applications such as language modelling, machine translation, sentiment analysis, and speech recognition. [14]

To understand how RNN works under the hood, an example of an NLP (Natural Language Processing) application named entity recognition is illustrated in Figure 5, this technique is used to detect names in a sentence:

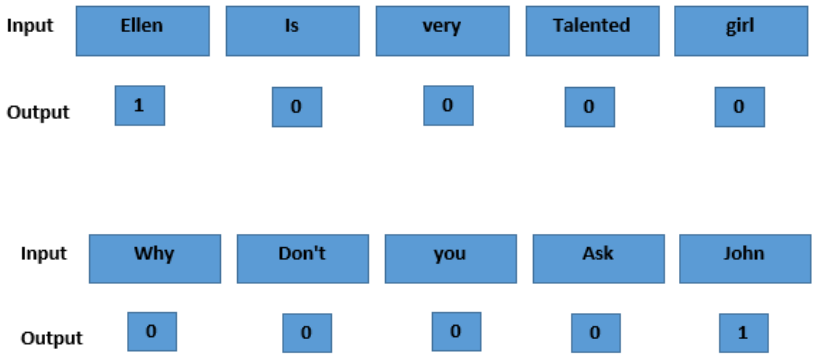


Figure 5. NLP application. Source: [5]

In the examples above, for each instance of training (sentence), we map each word with an output, if the word is named (john, Ellen …) we map it to 1. Otherwise, we map it to 0. So, to train RNN on sentences to recognize names within, the RNN architecture would be something like that in Figure 6.

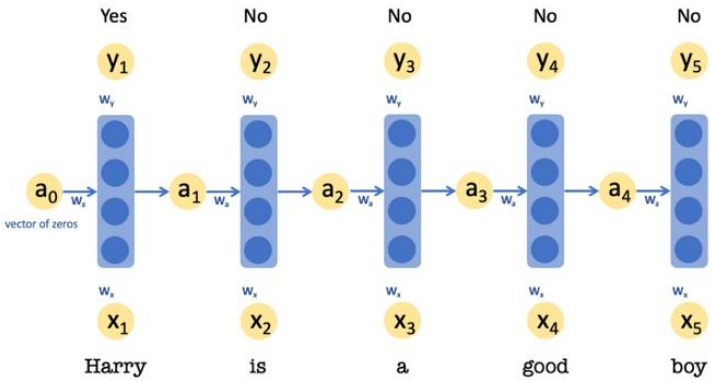


Figure 6. Example of RNN architecture. Source: [5]

Generative adversarial networks (GANS) consist of two neural networks at once: a generator that creates content and a discriminator that evaluates it.

The discriminator network receives training or generator-generated data. The degree of guessing by the discriminator of the data source further participates in the formation of the error. There is a competition between the generator and the discriminator: the first learns to deceive the second, and the second — to reveal the deception. It is difficult to train such networks because it is necessary not only to train each of them but also to adjust the balance between them. A typical application of GAN architectures is the stylization of photos, the creation of deep fakes, the generation of audio files, etc. [13]

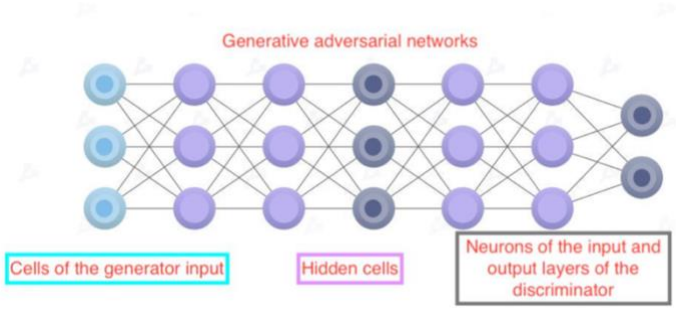


Figure 7. GAN. Source: [5]

3.4 Computer vision

3.4.1 Introduction to computer vision

Computer vision is a field that focuses on developing algorithms and mathematical techniques to enable computers to analyse images and videos, recognize objects, understand scenes, and infer relevant information from visual data. While humans can effortlessly perceive the three-dimensional structure of objects, interpret emotions from facial expressions, and effortlessly segment objects from a scene, replicating these abilities in computers remains a challenging task. Computer vision researchers have made significant progress in recovering three-dimensional shape and appearance from images. They have developed methods for computing 3D models of environments from photographs, creating dense 3D surface models, and even delineating objects and people in images to a certain extent. However, fully understanding images at the level of detail and causality that humans achieve remains an elusive goal. The difficulty of computer vision arises due to the inverse nature of the problem, where the goal is to recover unknowns based on insufficient information. Researchers use physics-based and probabilistic models or machine learning from large datasets to disambiguate potential solutions. Unlike modelling simple systems, such as the vocal tract for speech production, the visual world's complexity poses unique challenges. Computer vision relies on forward models from physics and computer graphics, which describe how objects move, how light interacts with surfaces, and how images are formed. In contrast, computer vision aims to reconstruct the world from images, describing properties like shape, illumination, and colour distributions. This inverse problem nature, coupled with the complex visual world, makes computer vision a challenging task. Despite the challenges, computer vision has found numerous practical applications across various domains. Some examples include:

1. Optical Character Recognition (OCR) for reading handwritten postal codes and number plate recognition.
2. Machine inspection for quality assurance in manufacturing, using stereo vision and X-ray imaging.
3. Retail applications like object recognition for automated checkout and fully automated stores.
4. Warehouse logistics with autonomous package delivery and robotic parts picking.

5. Medical imaging for registering pre-operative and intra-operative imagery and studying brain morphology.
6. Self-driving vehicles capable of driving point-to-point and autonomous flight.
7. 3D model building using photogrammetry for constructing 3D models from aerial photographs.

Additionally, computer vision has consumer-level applications, such as image stitching, exposure bracketing, morphing, 3D modelling, video stabilization, face detection, and visual authentication. The combination of engineering and scientific approaches, along with statistical techniques, allows for the formulation and solving of complex vision problems. The emphasis on algorithms that are robust to noise and efficient ensures practicality in real-world scenarios. Computer vision continues to advance rapidly, with a wide range of real-world applications and consumer-level possibilities, shaping the way we interact with visual data and revolutionizing various industries. As a result, it remains an exciting area of research and development for the future. [7]

3.4.2 Development of computer vision

Computer vision has undergone significant development over the years, evolving from its early beginnings in the 1970s to its current state in the 2010s. In the early days, computer vision was viewed as a way to mimic human intelligence and equip robots with intelligent behaviour. Researchers believed that solving the "visual input" problem would be a straightforward step towards tackling more complex challenges like higher-level reasoning and planning. One of the key differentiators of computer vision from digital image processing was its emphasis on recovering three-dimensional structures from images to achieve a comprehensive understanding of scenes. Early attempts involved extracting edges and inferring 3D structures from 2D lines. Various algorithms for line labelling and edge detection were developed during this period. In the 1980s, computer vision saw advancements in mathematical techniques for quantitative image and scene analysis. Image pyramids and wavelets were widely used for tasks such as image blending and correspondence search. Researchers also explored shape-from-X techniques, which included shape from shading, photometric stereo, and shape from texture. The concept of variational optimization and Markov random fields became prevalent in addressing complex vision problems. The 1990s witnessed the emergence of projective invariants for recognition and factorization techniques for structure-from-motion problems. Physics-

based vision and optical flow methods were further improved. Moreover, researchers started using statistical learning techniques, particularly for face recognition and curve tracking. In the 2000s, there was a notable shift towards data-driven and learning approaches in computer vision. Computational photography techniques, such as image stitching, HDR imaging, texture synthesis, and inpainting, gained prominence. Feature-based techniques combined with machine learning became essential for object recognition and scene understanding tasks. The 2010s marked a revolution in computer vision with the widespread adoption of large labelled datasets and deep learning techniques. Deep neural networks, especially convolutional architectures, dominated recognition and semantic segmentation tasks. Computational photography and vision algorithms found extensive applications in smartphones, enabling features like panoramic image stitching, high dynamic range imaging, and real-time augmented reality. Most of the existing computer vision systems were created based on ImageNet. But they still contained a lot of errors. Everything changed in 2012, when the AlexNet model, which used ImageNet, significantly reduced the error rate in image recognition, opening up the modern field of computer vision. [12]

Computer vision has come a long way from its early aspirations to mimic human intelligence. The field has evolved through the decades, leveraging mathematical techniques, statistical learning, and most significantly, the advent of deep learning. These advancements have enabled computer vision to find widespread applications in various industries, from smartphones to autonomous vehicles, transforming the way we interact with and understand visual data. [7]

3.4.3 Description of computer vision tasks

Computer vision is a dynamic and rapidly evolving field within computer science and artificial intelligence. It revolves around the development of algorithms and techniques that enable machines to interpret and understand visual information from the world around them. Among the fundamental tasks in computer vision, three key areas stand out: image classification, object detection, and image segmentation. These tasks play a crucial role in enabling machines to analyse and process visual data, ranging from simple image categorization to complex scene understanding. With advancements in deep learning and the availability of large labelled datasets, computer vision has witnessed significant progress in recent years, achieving remarkable results in various real-world applications.

1. Image classification

Image classification in computer vision has seen significant evolution over time. It began with traditional approaches like the "bag of words" method illustrated in Figure 8., which treated images as collections of visual words extracted from key points using descriptors like Scale-Invariant Feature Transform. Machine learning techniques, such as support vector machines, were used for classification. Another class of algorithms focused on part-based models, identifying objects by their constituent parts and geometric relationships, often using pictorial structures and trees. Part-based models found applications in face recognition, pedestrian detection, and pose estimation. Deep neural networks brought a revolution to image classification. They outperform traditional feature-based methods, especially with large labeled datasets. Context and scene understanding have further improved image classification by considering spatial relationships between objects. These algorithms enhance recognition accuracy, refine object detections, infer object locations, and predict object presence based on scene context.

In summary, image classification has evolved from classic feature-based methods to powerful deep neural networks, with context and scene understanding enhancing real-world applications. [7]

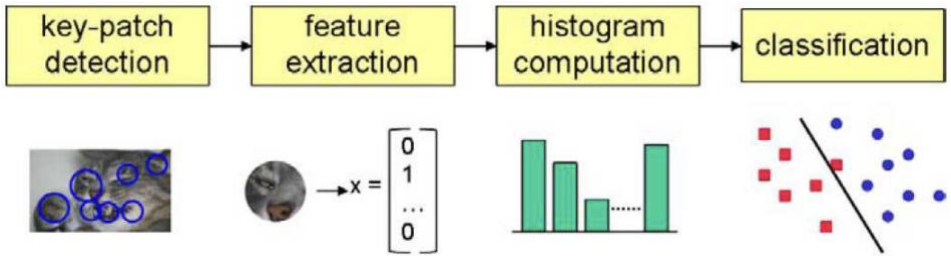


Figure 8. A typical processing pipeline for a bag-of-words category recognition system. Source: [7]

2. Object detection methods

In the realm of computer vision, modern object detection has undergone remarkable advancements through the integration of cutting-edge techniques. This journey of innovation has led to the development of powerful object detection methods, each with its unique approach. Among these methods, Convolutional Neural Networks (CNNs), YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector) stand as prominent pillars, shaping the landscape of object detection.

R-CNN: Region-based Convolutional Network (R-CNN), represents an early breakthrough in object detection with neural networks. It follows a two-stage process, commencing with the selective search algorithm to propose potential regions of interest. It works by first generating a set of potential object regions, extracting features from each region using a CNN, and then classifying and refining the bounding boxes for the predicted object categories, illustrated in Figure 9. Approximately 2,000 region proposals are extracted and then resized to a uniform size before undergoing classification using a neural network such as AlexNet [12]. This method evolves in subsequent versions like Fast R-CNN and Faster R-CNN, enhancing training and testing efficiency while significantly improving detection accuracy. The Faster R-CNN system replaces the slow selective search with a more efficient convolutional region proposal network (RPN), contributing to expedited inference. Notably, the introduction of a Feature Pyramid Network (FPN) further advances scale invariance, allowing for more robust detection across different object sizes. [6]

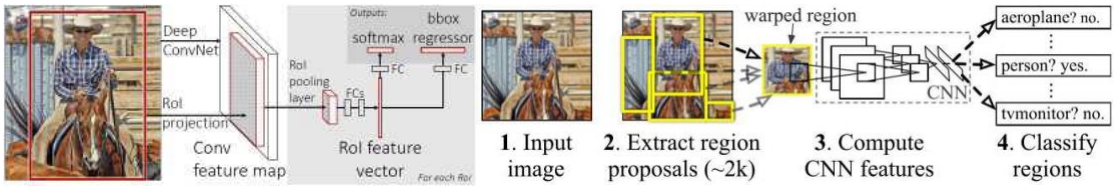


Figure 9. The R-CNN and Fast R-CNN object detectors. Source: [7]

YOLO: You Only Look Once (YOLO) is a family of object detection algorithms known for their speed and accuracy. They are well-suited for real-time applications. YOLO uses a single-stage detector, enabling it to identify and classify objects in a single pass of the input image. This makes YOLO significantly faster than other algorithms, but it can also come at the expense of accuracy.

The overview of how YOLO works:

1. Image Preprocessing: The input image is resized to a standard size, for example, 448x448 pixels. This is done to make the image compatible with the YOLO network architecture.

2. Feature Extraction: The input image is passed through a convolutional neural network (CNN) to extract features. The CNN extracts features from the image that are relevant for object detection.

3. Output Layer Predictions: The output layer of the CNN produces a prediction for each cell in a grid that is imposed on the input image. For each grid cell, the model estimates the bounding box coordinates (x, y coordinates, width, and height) along with a confidence score. Additionally, a class prediction is assigned to each grid cell.
4. Bounding Box Refinement: The bounding boxes and class probabilities from the output layer are refined using techniques such as non-maximum suppression (NMS). NMS removes overlapping bounding boxes and ensures that only the most confident detections are kept.
5. Object Detection: The final output of YOLO is a set of bounding boxes and class probabilities for all objects detected in the image.

The difference between YOLO and R-CNN algorithms was described in the paper by Redmon et.al [20]. Here are the reports of the summary of the accuracy and speed of R-CNN and YOLO in the following mean average precision scores on the PASCAL VOC 2007 dataset:

- RCNN: 62.4%
- YOLO: 44.1%

The paper describes speed as the frames per second (FPS) that an object detection algorithm can achieve. FPS is a measure of how fast an algorithm can process images and make detections. A higher FPS indicates faster speed.
Speed:

- RCNN: 7 FPS
- YOLO: 45 FPS

Looking at the paper research, the YOLO method is significantly faster than the RCNN method, while only sacrificing a moderate amount of accuracy. This makes YOLO a good choice for UI detection applications where real-time detection is critical.

YOLO offers significant advantages in terms of speed, but it has limitations, including lower accuracy for small or occluded objects and sensitivity to noise in the input image. Despite these limitations, YOLO remains a popular object detection algorithm due to its real-time capabilities and is likely to continue to be used extensively in various applications. It is used in a wide variety of applications, including self-driving cars, video surveillance, robotics, augmented reality, and virtual reality.

YOLO has evolved through several versions, each with its strengths and weaknesses. The most recent version is YOLOv8, released in 2023, and is the fastest and most accurate model to date. [15]

SSD: The Single Shot MultiBox Detector (SSD) represents another single-stage object detection technique, akin to YOLO. However, SSD uses a more complex architecture than YOLO, as illustrated in Figure 10. This means it can be less resource-efficient and more difficult to learn and set up.

SSD involves dividing an image into a grid and employing a set of predefined default boxes (anchors) with diverse aspect ratios to predict object locations and class scores. This method is tailored to handle objects of varying sizes efficiently. By integrating multiple convolutional layers with different scales in its architecture, SSD captures object information at multiple resolutions, aiding in object recognition across different contexts. This multi-scale approach contributes to the effectiveness of SSD in detecting objects with improved accuracy and speed. [8]

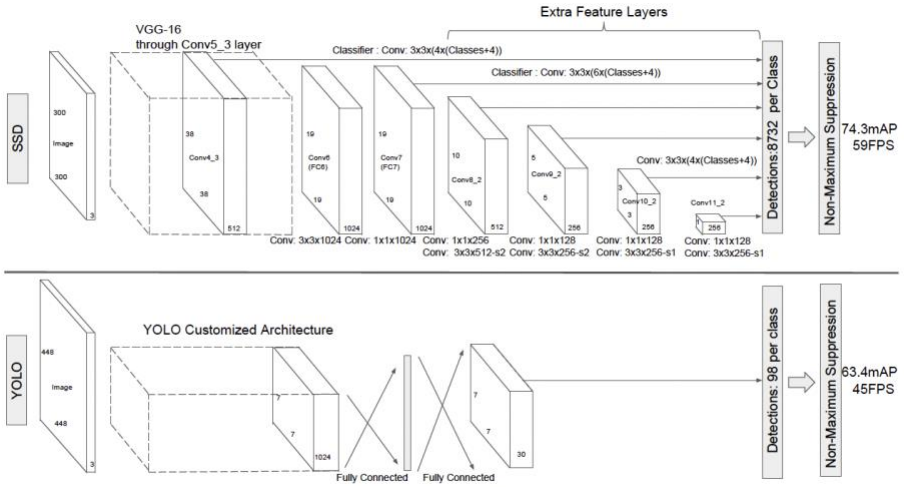


Figure 10. A comparison between two single shot detection models: SSD and YOLO. Source: [8]

3. Image segmentation

Image segmentation represents an advanced expansion of the concept of object detection. In this technique, the identification of objects within an image is taken to a more intricate level. Instead of merely outlining objects with bounding boxes, image segmentation involves creating pixel-wise masks that accurately define the boundaries of each object. This level of detail aids in ascertaining the shape of individual objects in a much finer manner. This approach is particularly beneficial in specialized areas like

medical image analysis and satellite imagery interpretation. In recent times, a multitude of methodologies for image segmentation have emerged. Among these, one of the notable techniques is Mask R-CNN, proposed by K He and collaborators in 2017. [16]

Image segmentation can be categorized into three types:

1. Instance Segmentation is a computer vision technique that detects and segments individual objects in an image. It goes beyond basic object detection by providing a more detailed understanding of the scene, including the precise location and boundaries of each object. This is achieved by generating pixel-level masks for each object instance.

Instance segmentation has a wide range of applications, including medical imaging, robotics, and autonomous vehicles. It is also used in many other fields, such as agriculture, retail, and security. [9]

2. Semantic Segmentation groups pixels in an image into common classes, assigning each pixel a label representing its object category. Unlike instance segmentation, which identifies separate instances, semantic segmentation highlights the distribution of different object categories by grouping pixels. This technique is crucial in various applications:

- Scene Understanding: Used in urban planning, virtual reality, and environmental monitoring for understanding layout and content.
- Autonomous Driving: Identifies lanes, pedestrians, signs, and vehicles, aiding navigation in self-driving cars.
- Medical Imaging: Segments organs and structures, assisting in diagnosis and treatment planning.
- Object Detection: Defines regions of interest, serving as a precursor to object detection.
- Agriculture: Monitors crops and land cover for improved agricultural practices.
- Augmented Reality: Aligns virtual objects with real scenes, enhancing augmented reality experiences.

While lacking instance-specific details, semantic segmentation forms the foundation for advanced techniques, contributing to diverse applications across industries. [9]

3. Panoptic Segmentation merges semantic and instance segmentation, aiming to comprehensively understand visual scenes. In semantic segmentation, each pixel is labeled with its semantic category, while instance segmentation associates pixels

with specific objects, providing pixel-accurate masks for each object. The goal of panoptic segmentation is to accurately segment and classify all objects and assign labels to contextual elements. Progress in this field includes a notable metric by Kirillov [9], evaluated across diverse datasets and compared to human consistency, with the COCO dataset expanding its scope to include panoptic segmentation assessment. This approach, combining precision from instance segmentation and broader context from semantic segmentation, decodes intricate object relationships, offering valuable insights for scene analysis and autonomous systems. [9]

Figure 11 shows image segmentation types.

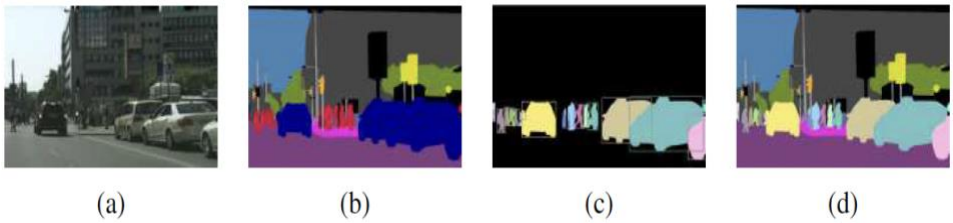


Figure 11. Examples of image (a) original image; (b) semantic segmentation, (c) instance segmentation, and (d) panoptic segmentation. Source: [9]

3.4.4 Other object detection solutions

The paper "Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination?" by Chen et al. (2022) [10] solves the problem of UI (user interface) detection by proposing a hybrid method that combines traditional image processing methods and deep learning methods. The paper's proposed method first pre-processes the GUI image using a traditional edge detection algorithm. This helps to remove noise from the image and makes it easier for the deep learning model to detect the UI elements.

The pre-processed GUI image is then fed to a deep-learning model to generate candidate UI element boxes. These candidate boxes are then filtered using a traditional color segmentation algorithm to remove false positives.

Finally, the filtered UI element boxes are classified using a deep learning model to identify the type of UI element in each box.

The paper's proposed hybrid method outperforms both traditional image processing methods and deep learning methods on a variety of UI element detection datasets. This

suggests that the hybrid method can solve the problem of UI detection more effectively than either traditional image processing methods or deep learning methods alone.

Here is a summary of how the paper's proposed hybrid method solves the problem of UI detection:

- Pre-processing the GUI image helps to remove noise and make it easier for the deep learning model to detect the UI elements.
- Generating candidate UI element boxes helps to identify potential UI elements in the image.
- Filtering the candidate UI element boxes using colour segmentation helps to remove false positives.
- Classifying the filtered UI element boxes using a deep learning model helps to identify the type of UI element in each box.

Overall, the paper's proposed hybrid method solves the problem of UI detection by combining the strengths of traditional image processing methods and deep learning methods. [10]

3.4.5 Publicly available different datasets

In the field of computer vision, the choice of datasets plays a pivotal role in model development and evaluation. To prepare a dataset, you can use one of two ways: create a custom dataset according to specific needs, or use one of the many publicly available datasets. Here are some of the well-known datasets of images with user interface elements.

The COCO dataset is a big collection of pictures for computer vision tasks. It has over 200,000 images of 80 different objects, and each image has detailed labels that say where the objects are and what they are. COCO is very useful for tasks like finding object detection and classification. It has also been used to create competitions for computer vision, which has helped to improve the field. Scientists and programmers use COCO for many different things, so it is a very important dataset for object recognition and scene understanding. [17]

Rico stands out as one of the most extensive mobile UI datasets, designed to support a range of data-driven applications, including design search, UI layout generation, UI code generation, user interaction modelling, and user perception prediction. The Rico dataset includes over 10,000 applications and 70,000 screenshots. It covers 27 different categories. [18]

The ReDraw dataset is a collection of Android screenshots, GUI metadata, and labelled images of GUI components. It is designed to support machine learning-based tools for GUI prototyping and refactoring. The dataset includes over 14,000 screenshots, metadata for each screenshot, and over 190,000 of GUI components. It can be used for various tasks, including GUI element detection, layout generation, code generation, user interaction modelling and user perception prediction. [19]

4 Practical Part

In this practical part of the bachelor's thesis, the detection of user interface elements (UI) on web pages is investigated using the YOLO version 8 nano (YOLOv8n) and small (YOLOv8s) object detection models. YOLOv8 was chosen due to its recent release and the lack of extensive research on its application in UI detection. By comparing the smaller YOLOv8n with the slightly larger YOLOv8s, the impact of model size on detection accuracy and inference speed is evaluated. This comparison may help in understanding how model size affects resource utilization and performance trade-offs.

A publicly available experimental dataset of images obtained from Roboflow, a computer vision development environment, is used for training and evaluating models. The dataset consists of images of user interface elements, including buttons, checkboxes, text fields, etc.

The subsequent workflow includes model training, validation, and testing. The ultimate goal is to test the performance of the models and use them to detect user interface elements on the website images.

4.1 Datasets

The public experimental image dataset was obtained from Roboflow, a computer vision development environment. The dataset contains unique images, consisting of user interface elements, and their variations. They were created using simple transformation techniques, such as rotation and mirroring to enhance the accuracy and adaptability of detection models. Such variations extend the model's capabilities to recognize elements regardless of their orientation, which is crucial for real-world applications where user interface elements may appear in various orientations. The example of images is illustrated in Figure 12.

The dataset is collected mainly from images of applications for mobile devices, and screenshots of website interfaces. To facilitate data organization and access, a configuration file named *'data yaml'* is used. The file provides information about training, validation, and testing subsets directories. The total number of object classes is 14: 'Button', 'CheckBox', 'CheckedTextView', 'EditText', 'EditText-' 'ImageButton', 'ImageView', 'ProgressBar', 'RadioButton', 'RatingBar', 'SeekBar', 'Spinner', 'Switch',

'TextView'. Three subsets of the dataset: *train*, *valid*, and *test* contain 2086, 200, and 100 files respectively, and are used to train, validate, and test the neural network model.

Each subset has files with images in the *jpg.* format containing one or more user interface elements. There is another text file in the *txt.* format for each image with bounding box coordinates for each user interface object and corresponding class.

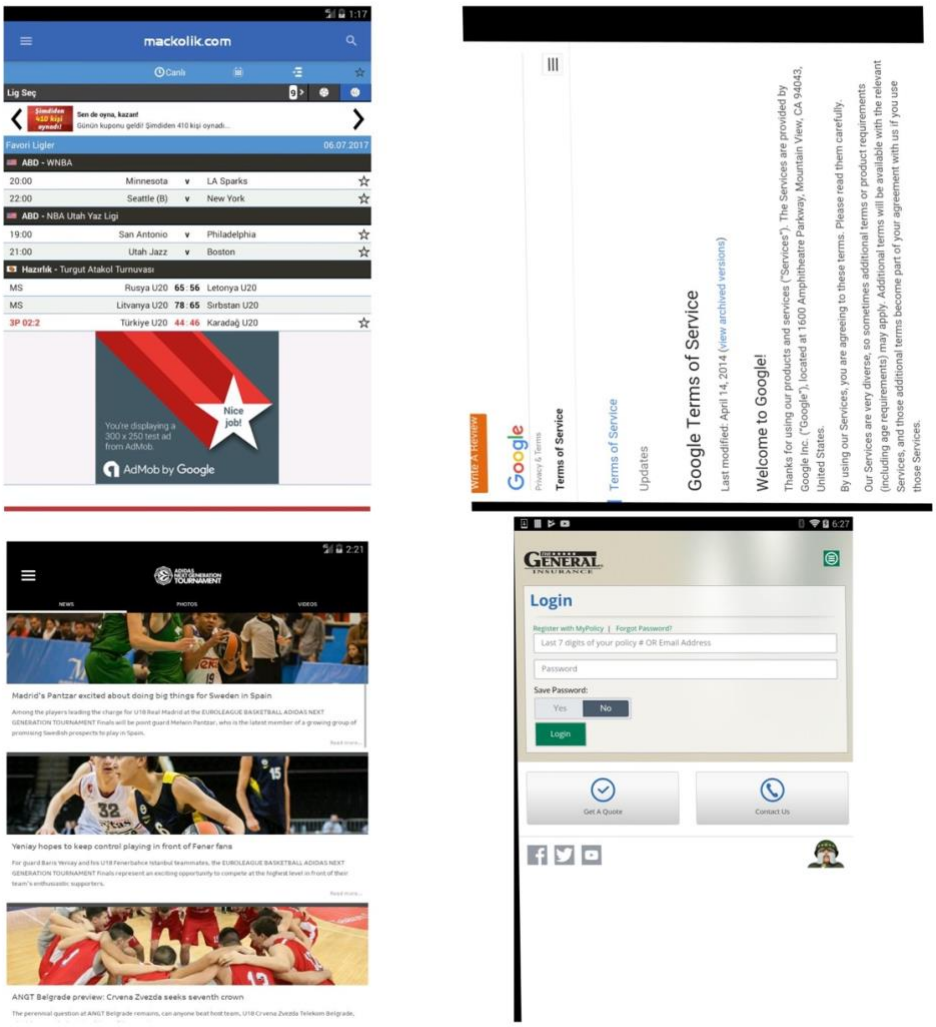


Figure 12. Images from the dataset.

4.2 Practical use of YOLOv8n and YOLOv8s

An open-source machine learning framework *PyTorch* was chosen for further work with detection of the user interface elements on the image. The models YOLOv8n and YOLOv8s training and implementation were done on the same dataset and parameters.

First, the YOLOv8n model was taken to work with. The pre-trained *yolov8n.pt* weights were obtained from the open-source *Ultralytics* and loaded into the framework. The first step was the model training on the training subset. The data from the set was loaded from the *data.yaml* file. Hyperparameters were set as default by *Ultralytics* as image size 480, and batch size 16. The model was trained on the duration of 200 epochs.

The next step was to validate model accuracy after it had been trained in the previous step. In this mode, the model is evaluated on a validation set to measure its accuracy and generalization performance. The subset *valid* is used for validation. After the training, the validation of the trained models was carried out again.
The third step is testing the trained model on a *test* dataset. The *test* set is needed to evaluate the performance of the model on data that it does not see during training. This allows us to make sure that the model will be able to work well on new data.

The next work was done with the YOLOv8s model. The pre-trained *yolov8s.pt* weights were obtained from the open-source *Ultralytics* and loaded into the framework. The training was done on the same dataset and adjusted hyperparameters as model YOLOv8n. Then the model was validated and tested.

After the YOLOv8n and YOLOv8s models were tested, they were used for the detection of the user interface elements webpage on the image of the chosen screenshot *'webscreen.jpg'* that is shown in Figure 13.

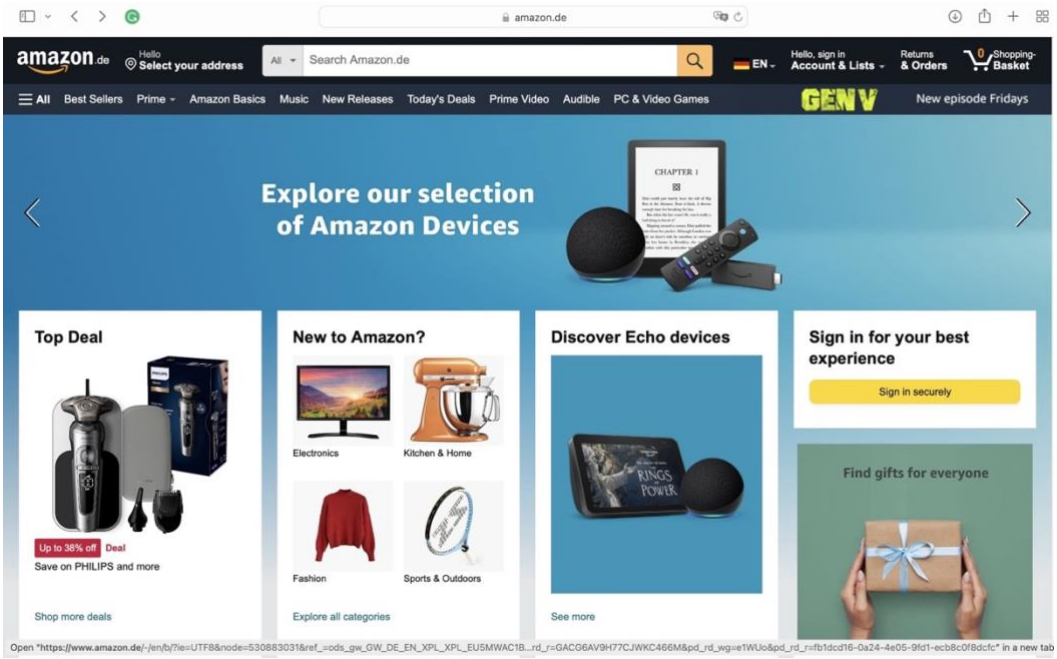
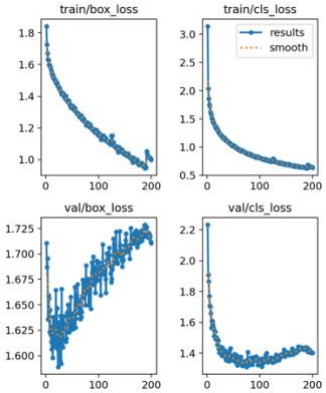


Figure 13. The webpage screenshot from the site “Amazon” https://www.amazon.de/.

5 Results and Discussion

5.1 Results of the practical part

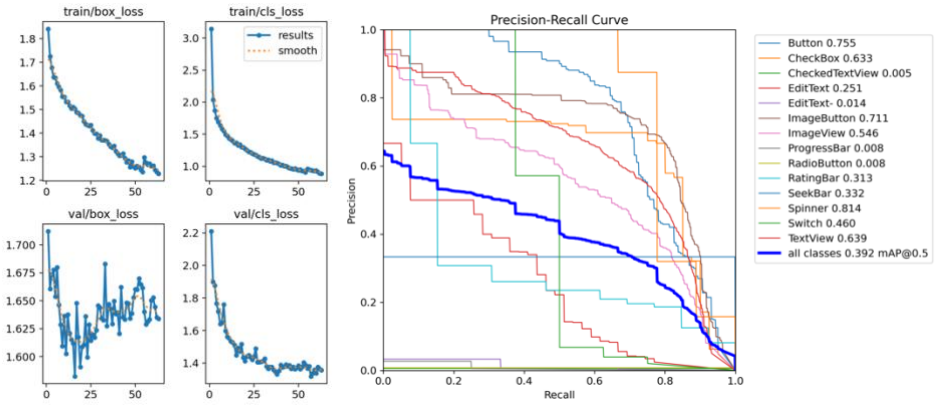
The model YOLOv8n has been trained at first for 200 epochs. The results of the validation on Graph 1. showed the model overfitting, meaning that it learns the training data too well, to the point where it cannot generalize to unseen data.



Graph 1. The result of validation and training of the YOLOv8n on 200 epochs.

Accordingly, the model YOLOv8n has been trained again with adjusted hyperparameters. The image size is 640 because a larger input image size gives the model more information to work with and improves the detection performance. The recommended value for a batch on a dataset with 2086 images using a 16-core graphics processor is 16, so this value was set. The number of epochs was set as 70 with a patience of 20, meaning that if in 20 epochs there is no improvement in training, it stops to prevent overfitting.

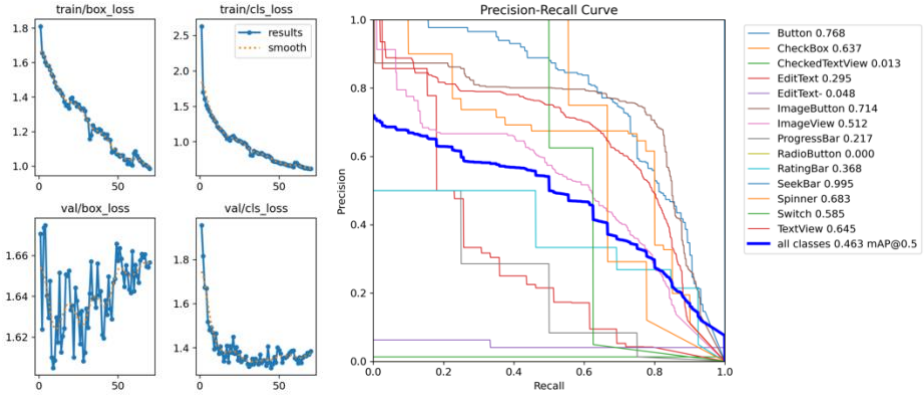
The early stop (patience) was triggered at epoch 63 and the training was stopped. The best result of the 43rd epoch was saved. The results of the training and validation are depicted in Graph 2.



Graph 2. The result of validation and training of the YOLOv8n.

The precision recall shows mean average precision (mAp) for all classes is 0.392 at a threshold of 0.5, which means that the model more accurately decomposes the user interface elements in the images. The val/box_loss value is higher than the train/box_loss, but it still decreases over time. This indicates that the model learns to better predict the bounding boxes of objects, but may to some extent override the training data.

The model YOLOv8s has been trained for 70 epochs. The results of the validation are in Graph 3. The val/box_loss value is higher than the train/box_loss, and there is a tendency for this value to decrease. The precision recall shows mean Average Precision (mAp) for all classes is 0.463 at a threshold of 0.5, which means that the model more accurately identifies the user interface elements in the images.



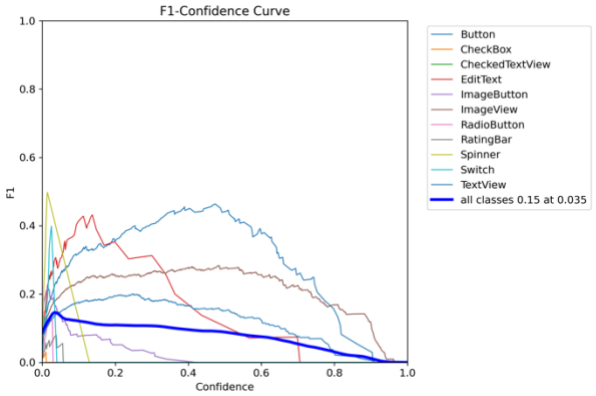
Graph 3. The result of validation and training of the YOLOv8s.

The F1 score is a metric that is used to evaluate the performance of a classification model. It is the mean of the model's precision and recall. Precision is the fraction of correct

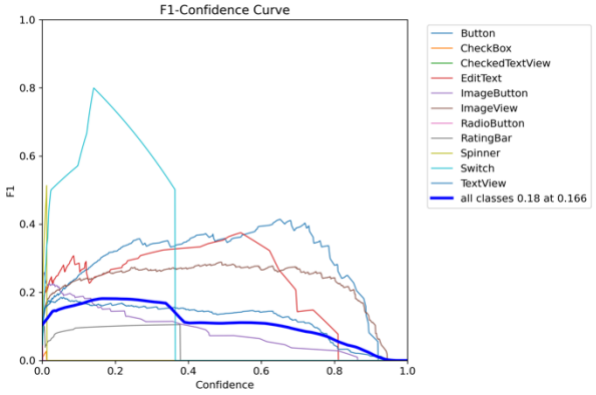
positive predictions, and recall is the fraction of actual positive cases that are correctly predicted.

The YOLOv8s model illustrated in Graph 5., has a higher F1 score (0.166) at a higher threshold than the YOLOv8n model F1 score (0.035), illustrated in Graph 4., suggests that it is better at detecting objects at higher confidence levels.

The fact that the first model was trained for more epochs than the second model has contributed to its higher F1 score. A longer training period allows the model to learn more about the data and to better generalize to new data.

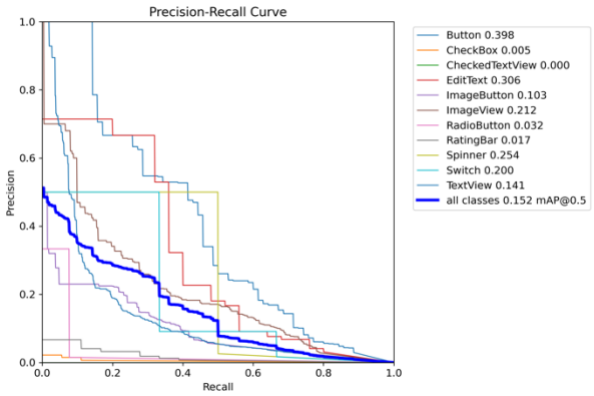


Graph 4. F1-confidence of YOLOv8n.



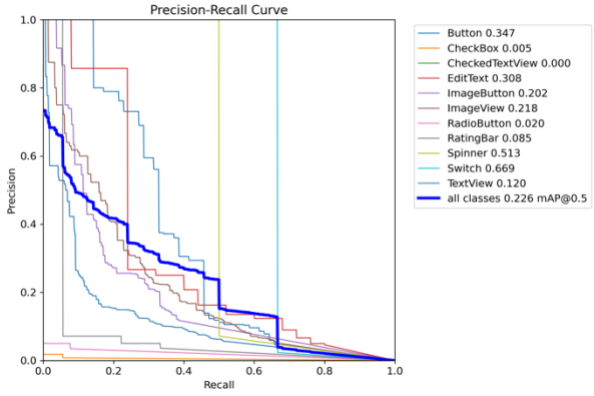
Graph 5. F1-confidence of YOLOv8s.

The precision and recall metrics from the YOLOv8n model testing are shown in Graph 6., and from the YOLOv8s model are shown in Graph 7. YOLOv8n: mAp is 0.152 at a threshold of 0.5, which is lower than it was after training mode, which means that the model is not able to generalize well to new data, perhaps because the test data may be more complex than the training data.



Graph 6. Precision-Recall (test), YOLOv8n.

YOLOv8s: mAp is 0.226 at a threshold of 0.5 is also lower in testing mode compared to training mode. The drop in mAp is not always a cause for concern. If the mAp in testing mode is still high enough for the model to be useful like in the YOLOv8s model, then the drop in mAp may be acceptable.



Graph 7. Precision-Recall (test), YOLOv8s.

The final step involved the detection of user interface elements in the image using both YOLOv8n and YOLOv8s models. The recognition results of YOLOv8n are shown in Figure 14, and those of YOLOv8s are shown in Figure 15. As expected, the results of recognizing user interface elements with YOLOv8n were less accurate compared to YOLOv8s. This is because the YOLOv8s model is more complex. This increased complexity allows the YOLOv8s model to learn more complex features and patterns in the training data, which can improve its generalization performance.

Metrics:	Model YOLOv8 small	Model YOLOv8 nano
mAP (training)	0.0.463 at threshold 0.5	0.392 at threshold 0.5
F1 (training)	0.18 at threshold 0.166	0.15 at threshold 0.035
mAP (testing)	0.226 at threshold 0.5	0.152 at threshold 0.5

Table 1. Metrics of small and nano models.

Table 1. compares the performance of two YOLOv8 models: nano and small, on the UI detection task. The table shows that the small model outperforms the nano model in terms of accuracy. It is slightly larger than the nano model and is a good choice for applications where accuracy is important. Nano is very fast to run, but it also sacrifices some accuracy. This model is a good choice for applications where speed is critical, but accuracy is not as important.

The work presented in this thesis has successfully demonstrated the effectiveness of YOLOv8 nano and YOLOv8 small object detection models for user interface detection on the images of webpages. Although these models show promising results, there is still room for further improvement and research. Several factors were found that impacted the work and could lead to improved recognition results in the future.

1. Investigate the use of hybrid models:

Hybrid methods refer to a combination of two or more different detection techniques to achieve a balance between speed and accuracy. These methods aim to leverage the strengths of each approach to overcome their individual limitations and provide a more comprehensive and robust detection solution.

2. Explore the use of transfer learning:

Transfer learning, a technique that utilizes a pre-trained model as a foundation for a new model, could prove to be a valuable tool in enhancing model performance. By fine-tuning a pre-trained object detection model, such as YOLOv8, on a dataset of user interface element images, transfer learning could significantly improve the model's ability to detect elements accurately.

3. Develop a more comprehensive UI detection dataset:

The current dataset used in this work is relatively small and does not cover the full range of UI elements that are found on web pages. Developing a more comprehensive

dataset would allow for more rigorous evaluation of UI detection models and would also help to identify any weaknesses in the models

Overall, this work demonstrates the importance of careful model selection and hyperparameter tuning to achieve optimal performance in object detection tasks.

Figure 14. Result of detection UI elements. YOLOv8n.

Figure 15. Result of detection UI elements. YOLOv8s.

5.2 Comparison of the work of Chen et al. (2022) and the current work on UI detection

Chen et al. (2022) [10] explored the use of deep learning for object detection in graphical user interfaces (GUIs). They compared the performance of traditional image processing methods with deep learning models and found that deep learning models outperformed traditional methods on a variety of GUI detection tasks.

In the current work, two YOLOv8 models were trained for GUI detection: YOLOv8 nano and small. The small model outperformed the nano model in terms of mean average precision (mAP) and F1 score. The early stopping regularization technique was used in the work to prevent overfitting, while Chen et al. did not mention using any regularization techniques in their work.

Chen et al. (2022) used a larger dataset of 10,000 images, while the current work used a smaller dataset of 2,086 images. This difference in dataset size may have contributed to the better performance of the models in the work of Chen et al. compared to the current work. Larger datasets typically provide more information for deep learning models to learn from, allowing them to capture more complex patterns and features. This can lead to improved generalization performance, meaning that the models are better able to perform on new data that they have not been trained on.

Overall, the findings of the two works are consistent: deep learning models are effective for GUI detection.

6 Conclusion

User interface (UI) element detection focuses on identifying and localizing UI components within digital interfaces. It is essential for various applications, including accessibility, automated testing, and the development of intelligent systems that can interact with graphical user interfaces. As technology continues to advance, UI detection methods are evolving to enhance accuracy, speed, and adaptability across different user interfaces and platforms.

The described approach to detect UI elements, in the thesis, is machine learning. It uses artificial neural networks that are trained on large datasets that contain images with user interface elements. These images are gathered from websites and mobile applications. Three different object methods are described: R-CNN, YOLO, and SSD. YOLO and SSD are commonly used in UI detection tasks due to their ability to efficiently locate and classify multiple objects in an image. The YOLO algorithm has a less complex architecture than SSD which makes it easier to use and is faster than the R-CNN. YOLO excels at processing images quickly, making it suitable for real-time applications.

In the practical part, two sizes of the latest version of the YOLO model are selected and evaluated. A publicly available dataset is used to train and evaluate these models to detect interface elements in webpage images. The performance of YOLO models is evaluated using precision, recall, and confidence metrics. The small model is better in terms of accuracy and the nano model is faster. However, the results of the comparison of using both models in UI detection on the image show relatively small differences between them. The choice between the nano and the small models depends on the specific application requirements and the desired balance between accuracy and speed. For applications that prioritize real-time performance, the nano model is a better choice, while for applications that require higher accuracy the better is the small model.

Based on the practical work results and their comparison with another work, more comprehensive datasets of UI elements can be used for enhancing model performance. The combination of multiple detection techniques may help to achieve a favorable trade-off between processing speed and detection accuracy

7 References

1] RUSSEL, S., NORVIG, P. *Artificial Intelligence: A modern approach*. 4th edition. 2021. ISBN: 9780134610993.

[2] GERON, A. *Hands-on machine learning with sci-kit-learn, Keras, and tensorflow: Concepts, tools, and techniques to build Intelligent Systems*. 2nd edition. 2019. O'Reilly. ISBN: 9781492032649

[3] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep learning*. 2016. ISBN: 9780262035613

[4] BEYSOLOW, T. II. *Introduction to deep learning using r: A step-by-step guide to learning and implementing deep learning models using R*. 2017. ISBN: 9781484227336

[5] BOUFELOUSSEN, O. *Simple explanation of recurrent neural network (RNN)*. 2020. Available at: https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363 (Accessed: 11 October 2023).

[6] ODEMAKINDE, E. *Everything about mask R-CNN: A beginner's guide*. 2022. Available at: https://viso.ai/deep-learning/mask-r-cnn/ (Accessed: 11 October 2023).

[7] SZELISKI, R. *Computer vision: Algorithms and applications, 2nd ed.* 2021. Available at: https://szeliski.org/Book/ (Accessed: 11 October 2023). ISBN: 9783030343712

[8] LIU, W. *et al.SSD: Single shot multibox detector, arXiv.org*. 2016. Available at: https://arxiv.org/abs/1512.02325 (Accessed: 05 August 2023).

[9] KIRILLOV, A. *et al. Panoptic segmentatio, arXiv.org*. 2019. Available at: https://arxiv.org/abs/1801.00868 (Accessed: 10 September 2023).

[10] CHEN, J. *Object detection for graphical user interface: Old Fashioned or deep Learning or a Combination? arXiv.org*. 2022. Available at: https://arxiv.org/pdf/2008.05132.pdf (Accessed: 05 November 2023).

[11] HIMANSHU, S. *Practical machine learning and image processing*. 2019. Apress. ISBN: 978-1-4842-4149-3.

[12] KRIZHEVSKY, A., SUTSKEVER, I., HINTON, G.E. *ImageNet classification with deep convolutional Neural Networks*. 2012. Available at: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (Accessed: 25 November 2023).

[13] XIAOYUE, J., HADID, A. *Deep learning in object detection and recognition*. 2019. ISBN: 9789811051517.

[14] KURKOVA, V., MANOLOPOULUS, Y., HAMMER, B. *Artificial Neural Networks and Machine Learning – ICANN.* 2018. ISBN: 978-3-030-014118-6.

[15] HUSSAIN, M. *Yolo-V1 to Yolo-V8, the rise of Yolo and its complementary nature toward digital manufacturing and industrial defect detection*. 2023. Available at: https://www.mdpi.com/2075-1702/11/7/677 (Accessed: 25 November 2023).

[16] HE, K. *et al. Mask R-CNN, arXiv.org*. 2018 Available at: https://arxiv.org/abs/1703.06870 (Accessed: 25 November 2023).

[17] LIN, T.-Y., MAIRE, M. and BELONGIE, S. *Microsoft Coco: Common Objects in Context, arXiv.org*. 2015. Available at: https://arxiv.org/abs/1405.0312 (Accessed: 25 November 2023).

[18] DEKA, B., HUANG, Z. and FRANZEN, C. *Rico: A Mobile app dataset for building data-driven design applications*. 2017. Available at: https://dl.acm.org/doi/pdf/10.1145/3126594.3126651 (Accessed: 25 November 2023).

[19] MORAN, K., BERNAL-CARDENAS, C. and CURCIO, M. *Machine learning-based prototyping of graphical user interfaces for mobile apps.* 2018. Available at: https://ieeexplore.ieee.org/document/8374985 (Accessed: 25 November 2023).

[20] REDMON, J. *et al. You only look once: Unified, real-time object detection*. 2016. Available at: https://arxiv.org/abs/1506.02640 (Accessed: 26 November 2023).

8 List of pictures, tables and graphs

8.1 List of pictures

8.2 List of tables

8.3 List of graphs