POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

AI-Powered Autonomous Industrial Monitoring: Integrating Robotics, Computer Vision, and Generative AI

Supervisors

Prof. Paolo GARZA

Dr. Oscar PISTAMIGLIO

Candidate

Enrico GIACALONE

March 2025

Summary

Innovations in Artificial Intelligence (AI), Robotics, and Computer Vision are revolutionizing industrial monitoring by automating traditionally manual and time-consuming tasks, increasing both efficiency and safety.

This thesis explores the integration of these technologies into a proprietary AI framework, to enhance industrial visual analysis through Autonomous Mobile Robots (AMRs), focusing on two key tasks:

1. **Automated gauge reading** addresses the challenge of continuously monitoring pressure gauges in large-scale industrial environments, a process traditionally performed by human operators and prone to errors and inefficiencies.

   To overcome these limitations, this study employs Boston Dynamics' Spot, a quadruped robot functioning as a mobile Internet of Things (IoT) platform capable of executing pre-configured inspection missions. Equipped with high-resolution cameras and LiDAR sensors, Spot captures images of both analog and digital gauges that are then analyzed using state-of-the-art AI models, including Optical Character Recognition (OCR), Computer Vision techniques, and Multimodal Large Language Models (LLMs).

   The study compares different approaches for extracting and interpreting gauge readings, highlighting the strengths and limitations of each model.

2. **Autonomous surveillance in restricted areas** focuses on autonomous surveillance, leveraging the DJI Matrice 3TD, a drone designed for industrial applications, integrated into an alarm system to conduct real-time inspections in restricted areas.

   By employing Computer Vision and Deep Learning models such as YOLO (You Only Look Once) for object detection, the system enables the drone to rapidly identify anomalies, unauthorized intrusions, and potential safety hazards with high accuracy.

   This section examines multiple analyses performed on the dataset used, aiming to optimize results based on key reference metrics.

Using a research-driven approach, all models and methodologies are evaluated based on quantitative metrics and real-world applicability.

The findings demonstrate significant improvements in efficiency, accuracy, and safety, contributing to the advancement of AI-driven industrial automation and monitoring.

Future research will focus on enhancing model robustness, potentially integrating multimodal sensor data, and expanding capabilities to additional industrial applications.

Table of Contents

List of Tables v

List of Figures vi

Acronyms ix

1 Introduction 1
1.1 Context and objective 1

2 Used technologies 4
2.1 Robotics 4
2.1.1 Boston Dynamics - Spot 5
2.1.2 DJI - Matrice 3TD 8
2.2 Computer Vision 10
2.2.1 Ultralytics - YOLO (You Only Look Once) 13
2.2.2 Ultralytics - SAM (Segment Anything Model) 17
2.3 Generative AI 20
2.3.1 OpenAI - Generative Pre-trained Transformer (GPT) 31

3 Proposed tasks 35
3.1 Reading of gauge measurements 35
3.1.1 Digital gauges 40
3.1.2 Analog gauges 50
3.2 Autonomous surveillance on restricted areas 59

4 Conclusion 65
4.1 Challenges and Opportunities 65

Bibliography 67

List of Tables

3.1 LLMs comparison on digital gauge reading task 48
3.2 Possible improvements on digital gauge reading task 49
3.3 Methodology comparison on analog gauge reading task 58
3.4 Results from the first train on VisDrone detection dataset 62
3.5 Results from the second train on VisDrone detection dataset 63
3.6 Results from the third train on VisDrone detection dataset 64

List of Figures

1.1 Equipment monitoring in industrial manual inspection 1
1.2 Sprint Reply logo 2
1.3 Reply research lab "Area42" 2

2.1 Human plant inspection 4
2.2 Boston Dynamics logo 5
2.3 Part of Boston Dynamics robots fleet 5
2.4 Boston Dynamics "Spot" during a routine plant inspection 6
2.5 Spot Base Robot Specifications 7
2.6 Spot Battery and Payload Specifications 7
2.7 DJI logo 8
2.8 DJI Matrice 3TD 8
2.9 An example of real world computer vision application 10
2.10 Image classification example 11
2.11 Object detection example 11
2.12 Instance segmentation example 12
2.13 Pose estimation example 12
2.14 Ultralytics logo 13
2.15 YOLO detection system. It (1) resizes the input image to 448x448, (2) runs a single CNN on it, and (3) thresholds the resulting detections by the model's confidence. 13
2.16 Non-Maximal Suppression (NMS) applied to a YOLO SxS grid . . . 14
2.17 YOLO original architecture 15
2.18 YOLO models performance comparison 16
2.19 SAM flexible promptable interface 17
2.20 SAM architecture 18
2.21 SAM 2 promptable video segmentation 18
2.22 SAM 2 architecture 19
2.23 "Théâtre D'opéra Spatial", an image made using Generative AI . . . 20
2.24 Discriminative (left) and Generative (right) models difference 21
2.25 Generative Adversarial Networks (GANs) workflow example 22

2.26 Recurrent Neural Network (RNN) standard architecture 22
2.27 Long short-term memory (LSTM) recurrent unit 23
2.28 Sequence-to-Sequence (Seq2Seq) model in language translation . . . 24
2.29 Example of the "Attention" mechanism 25
2.30 Transformer architecture 26
2.31 Attention and Multi-Head Attention layers 27
2.32 Bidirectional Encoder Representations from Transformers (BERT) model 29
2.33 Example on how an LLM works 30
2.34 Example on how a Multimodal LLM works 31
2.35 OpenAI logo 32
2.36 ChatGPT web interface 32
2.37 OpenAI o1 (or Strawberry) new scaling paradigm 34

3.1 An industrial machinery equipped with digital and analog gauges . 35
3.2 A chemical Continuous Stirred Tank Reactor (CSTR) 36
3.3 An example of autonomous mission configuration 37
3.4 Spot equipped with payloads 37
3.5 Spot's LIDAR mapping 38
3.6 Spot's docking station 38
3.7 Spot's autonomous gauge mission 39
3.8 Verification by human personnel of an anomalous measurement . . . 39
3.9 An example of digital gauge 40
3.10 Optical Character Recognition (OCR) of a label 40
3.11 Hugging Face Hub interface 42
3.12 LLaVA (Large Language and Vision Assistant) architecture 43
3.13 LLaVA-NeXT "AnyRes" technique 44
3.14 Generation of "interleaved" sequence of images and texts 45
3.15 Example of an image for the digital gauge dataset 46
3.16 Confusion matrix in classification 47
3.17 Image preprocessing for OCR improvement 49
3.18 Calibration phase in previous methodology 50
3.19 Measurement phase in previous methodology 51
3.20 Gauge detection and crop 52
3.21 Heatmap prediction and clustering 53
3.22 Notch detection and categorization 53
3.23 Ellipse fitting through keypoints 54
3.24 Needle pixels segmentation and line fitting 54
3.25 Detection and extraction of scale numbers and unit 55
3.26 Projection and interpolation of scale markers 56
3.27 Intersection of needle line and estimated ellipse 56

3.28 Example of an image for the analog gauge dataset 57
3.29 Example of surveillance on a region of interest (ROI) 59
3.30 DJI's hangar, Dock 2 60
3.31 Template of email triggering drone missions 61
3.32 A VisDrone dataset labelled instance 61
3.33 Distribution of instances across classes in VisDrone dataset 62
3.34 Validation curves in VisDrone dataset 63

Acronyms

AI
Artificial Intelligence

AMR
Autonomous Mobile Robot

CNN
Convolutional Neural Network

GPT
Generative Pre-trained Transformer

IoT
Internet of Things

LLM
Large Language Model

OCR
Optical Character Recognition

RNN
Recurrent Neural Network

VLM
Vision Language Model

YOLO
You Only Look Once

Chapter 1

Introduction

1.1 Context and objective

In recent years, advancements in Artificial Intelligence (AI), Robotics, and Computer Vision have significantly transformed the industrial sector, that requires continuous monitoring of equipment to ensure operational efficiency, prevent failures, and enhance workplace safety.

Traditionally, these tasks have been performed by human operators (Fig. 1.1); however, manual inspections are time-consuming, prone to human error, and inefficient in large-scale industrial environments.

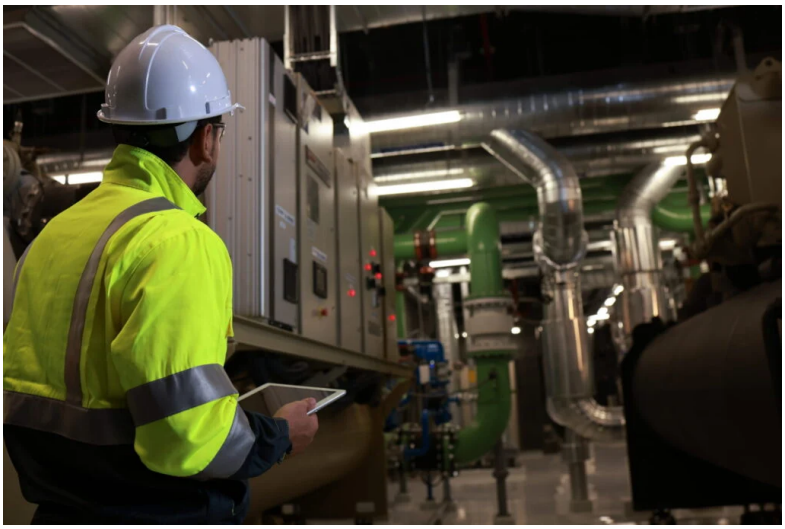

Figure 1.1: Equipment monitoring in industrial manual inspection

The increasing demand for automation in industrial monitoring has led to the

adoption of Autonomous Mobile Robots (AMRs) equipped with Computer Vision and AI-driven visual analysis: these technologies enable real-time inspection and enhanced data collection efficiency, also reducing the need for human intervention in potentially hazardous environments.

This thesis work is part of a project carried out by Sprint Reply [1] (Fig. 1.2), the IT consulting company within the Reply group that specializes in the implementation of Artificial Intelligence (AI) solutions, dedicated to optimizing business processes.

Figure 1.2: Sprint Reply logo

Thanks to them, I had the opportunity to carry out my activities within "Area42" [2] (Fig. 1.3), Reply's research center in Turin, where creative ideas can be explored and brought to life by leveraging the potential of cutting-edge technologies across various fields, including Industrial IoT, Autonomous Warehousing, Smart Mobility, Robotics, and the Metaverse.

Figure 1.3: Reply research lab "Area42"

In particular, the Robotics Lab, as a central component of my experience, is the

space where the aforementioned robot functionalities and opportunities are tested.

Here, other than directly providing consulting services to clients, Sprint Reply's objective is to operate as a product company, with the ultimate goal being experimentation and research.

Specifically the product is a proprietary AI framework, available as a web application built in Python, an object-oriented programming language.

Being modular, it is fully divided into components (modules) that operate independently within the system, each performing specific functions.

The concept behind this framework is to provide a set of useful "features" for various visual analysis applications in the industrial sector, integrating Robotics, Computer Vision, and Generative AI.

On its opening screen, the tabs "Streaming" and "Processing" are immediately visible, specifically:

- Streaming is the section of the framework that allows users to select a video source (this could be an integrated or external webcam, a video file, or one of the various components from the robot fleet, which we will discuss in detail in the next section) to analyze in real-time using Computer Vision algorithms.
- Processing is the section that takes a single image or video file as input, then performs an analysis using the aforementioned algorithms or with the application of Generative AI.

With reference to this framework, the objective of this work is to develop, evaluate and integrate new AI-powered features, focusing on two key tasks:

1. Automated Gauge Reading: Developing AI-based methodologies for reading both digital and analog gauge measurements
2. Autonomous Surveillance in Restricted Areas: Integrating a drone into an alarm system, to conduct real-time inspections in industrial facilities

To this end, some crucial steps were evaluating different AI models and frameworks, including Optical Character Recognition (OCR), Multimodal Large Language Models (LLMs), and other Computer Vision techniques within a real-world industrial setting.

Chapter 2 presents the related work in the form of an in-depth exploration of the technologies used, while Chapter 3 details all the steps followed during the execution of the proposed tasks.

Chapter 2

Used technologies

2.1 Robotics

In industrial facilities around the world, specialists carry out numerous control activities (Fig. 2.1) daily, including visual inspections of plants, verification of proper equipment functionality, and checking instruments like pressure gauges, thermometers, and valves.

The purpose of these inspections is to maintain the reliability and safety of all the components of the plant that workers regularly interact with.

Figure 2.1: Human plant inspection

However, what can be a monotonous, repetitive, and time-consuming task for

humans (often more than one person) can be easily automated with the help of modern autonomous mobile robots (AMRs).

Specifically, these robots enable the creation and execution of so-called "missions", predefined patrol routes within the facility, with customizable frequencies and strategic stops at designated locations.

During these inspections, the robots collect a vast amount of data in multiple formats, thanks to the numerous sensors and cameras they are equipped with. Their applications range from industrial plants, where they can detect abnormal temperatures or gas leaks, to natural disaster scenarios, where they can assist rescuers in locating survivors.

2.1.1 Boston Dynamics - Spot

A significant added value in my internship experience has been the partnership between my company and Boston Dynamics [3] (Fig. 2.2), with which we collaborate and regularly interact.

Figure 2.2: Boston Dynamics logo

It is an American engineering and robotics design company, global leader in developing and deploying highly mobile robots (Fig. 2.3), founded in 1992 as a spin-off from the Massachusetts Institute of Technology (MIT) [4].

Figure 2.3: Part of Boston Dynamics robots fleet

Headquartered in Waltham, Massachusetts, it was first acquired by Google in 2013 and then by SoftBank in 2017; it is now owned by the Hyundai Motor Group as of 2020.

Over the years, Boston Dynamics has developed a series of robots that, since 2019, have been made commercially available to reduce the dangerous, repetitive, and physically difficult aspects of work.

Among those, the one that best meets the diverse needs of clients in the industrial sector is Spot [5] (Fig. 2.4), the quadruped robot with canine-like features.

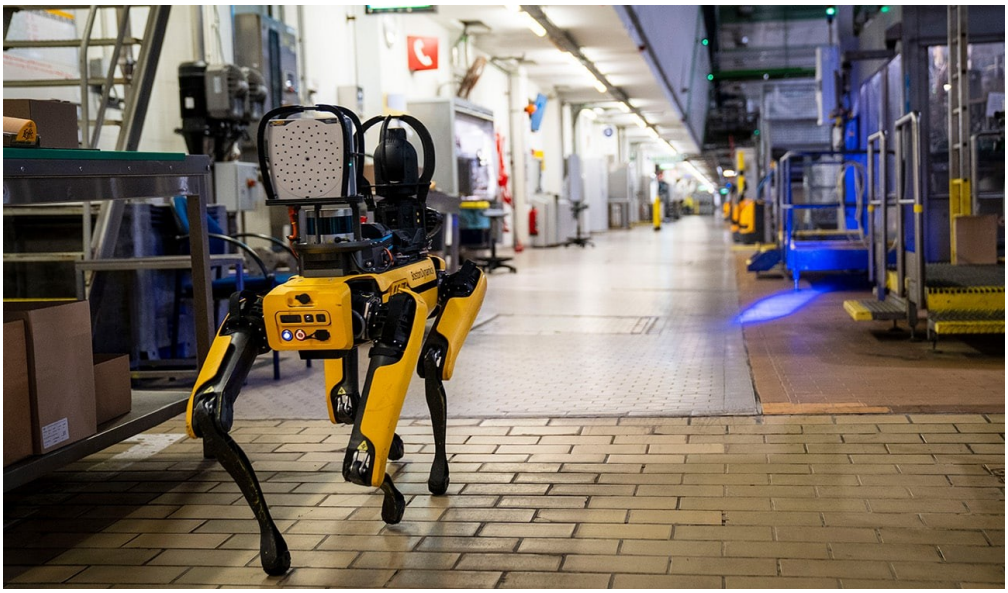

Figure 2.4: Boston Dynamics "Spot" during a routine plant inspection

It allows for the easy, precise, and safe automation of routine inspections and hazardous tasks for humans, thanks to remote control capabilities.

With a base cost of approximately $75,000, Spot can be considered a mobile, agile, and autonomous IoT station, highly customizable with numerous “payloads” such as thermal, panoramic, and high-resolution optical zoom cameras; temperature, humidity, acoustic, and gas sensors, all of which can trigger notifications if values exceed predefined ranges.

Equipped with integrated edge computing units, Spot conducts inspection rounds, accurately and frequently collecting large volumes of data via its various sensors. This data is then processed according to the use case or simply stored on a data platform for further analysis and review by specialists.

Designed for use on unstructured terrains or in restricted spaces, Spot is equipped with an obstacle avoidance system supported by its numerous cameras and integrated sensors. Moreover, its stability is robustly ensured by specific reinforcement

learning (RL) techniques.

Spot also has an IP54 protection rating, meaning it can withstand dust and rain but is not resistant to high-intensity sprays or immersion.

For more detailed information, refer to the specification tables below:

Base Robot **Battery & Payload**

DIMENSIONS

Length
1100 mm (43.3 in)

Width
500 mm (19.7 in)

Height (Sitting)
191 mm (7.5 in)

Default Height (Walking)
610 mm (24.0 in)

Max Height (Walking)
700 mm (27.6 in)

Min Height (Walking)
520 mm (20.5 in)

Net Mass/Weight (Spot with battery)
32.7 kg (72.1 lbs)

LOCOMOTION

Max Speed
1.6 m/s

Max Slope
±30°

Max Step Height
300 mm (11.8 in)

ENVIRONMENT

Ingress Protection
IP54

Operating Temp.
-20°C to 55°C

TERRAIN SENSING

Horizontal Field of View
360°

Range
4 m (13 ft)

Lighting
> 2 Lux

Collision avoidance
maintains set distance from stationary obstacles

CONNECTIVITY

WiFi
2.4GHz / 5GHz b/g/n

Ethernet

Figure 2.5: Spot Base Robot Specifications

Base Robot **Battery & Payload**

BATTERY

Battery Capacity
564 Wh

Average Runtime*
90 mins

Standby Time
180 mins

Recharge Time
60 mins

Length
324 mm (12.8 in)

Width
168 mm (6.6 in)

Height
93 mm (3.7 in)

Mass/Weight
5.2 kg (11.5 lbs)

*Runtime may vary depending on payloads and environmental factors

PAYLOAD MOUNTING

Max Weight
14 kg (30.9 lbs)

Mounting Area
850 mm (L) x 240 mm (W) x 270 mm (H)

Mounting Interface
M5 T-slot rails

Connector
DB25 (2 ports)

Power Supply
Unregulated DC 35-58.8V, 150W per port

Integration
Available software API and hardware interface control document

Figure 2.6: Spot Battery and Payload Specifications

2.1.2 DJI - Matrice 3TD

DJI, an acronym for Dà-Jiāng Innovations (in Chinese, “Great Frontier Innovations”), is a Chinese technology company (Fig. 2.7) headquartered in Shenzhen that designs and manufactures commercial unmanned aerial vehicles (UAVs), commonly known as drones, primarily for aerial photography and videography.

Figure 2.7: DJI logo

The company was founded in 2006 by Frank Wang, who began by selling flight control modules to the niche market of drone enthusiasts from his dormitory [6].

As of June 2024, DJI accounted for over 90% of the global consumer drone market, to the point that its technology is widely used across industries such as music, television, and film.

Among the numerous series of models produced by DJI, the "Matrice" series is specifically designed for industrial applications, including surveying, inspection, search and rescue, and firefighting.

In particular, I had the opportunity to work with the "Matrice 3TD" model [7] (Fig. 2.8), an advanced drone equipped with a wide-angle fisheye camera, a telephoto lens, and an infrared camera: these features enable the capture of high-quality, real-time images, significantly enhancing the precision of operations.

Figure 2.8: DJI Matrice 3TD

The Matrice 3TD allows fully autonomous route configuration with complete safety, thanks to the six-direction collision avoidance system it is equipped with.

The system's advanced sensors can analyze the surrounding environment even before takeoff, ensuring an accurate site assessment.

The drone boasts a maximum flight time of 50 minutes and an operational range of up to 10 kilometers; it is also equipped with dual antennas with RTK module for centimeter-level precise localization of the RTH (Return To Home) position.

Additionally, the Matrice 3TD has an IP55 certification, making it resistant to dust and water, which is ideal for operation in challenging environmental conditions.

2.2 Computer Vision

Computer vision is an interdisciplinary field of artificial intelligence that leverages machine learning and neural networks to teach computers and systems to derive meaningful information from digital images, videos, and other visual inputs [8].

One might say that if AI enables computers to think, computer vision enables them to see, observe, and understand the world as humans do.

Generally, these methods involve an initial acquisition phase, followed by processing, and ultimately analysis and comprehension of the input, made possible by extracting high-dimensional numerical or symbolic information.

Within computer vision, there are numerous tasks [9], each with a specific goal and use case, that can be applied across various fields, from autonomous driving (Fig. 2.9) to defect detection, medical diagnosis, and video surveillance.

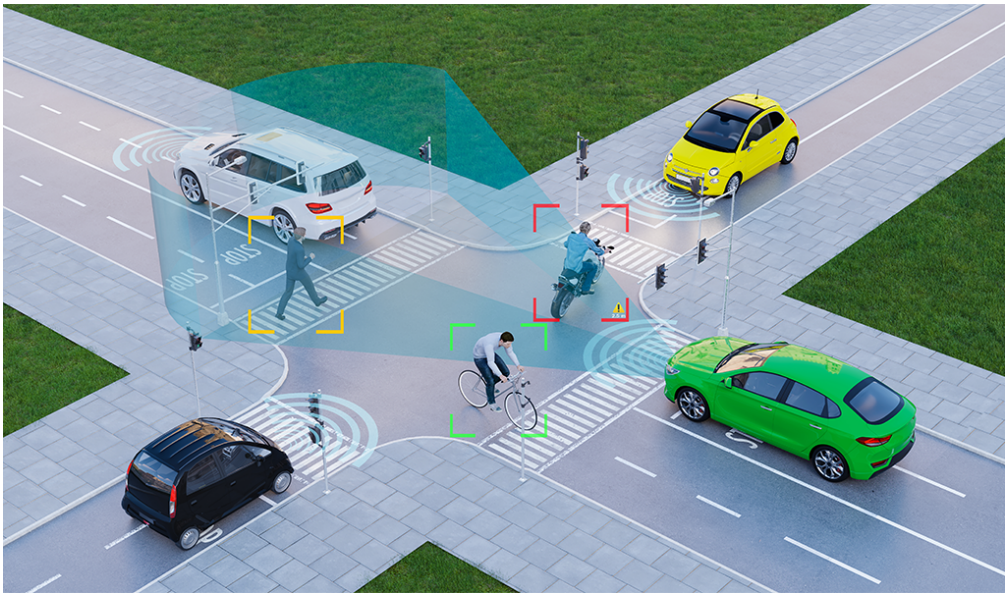

Figure 2.9: An example of real world computer vision application

Image Classification

The simplest task is undoubtedly image classification, as it involves classifying the entire image into one or more predefined classes based on its content.

The output of an image classifier consists of the class label(s) (Fig. 2.10), optionally followed by a confidence score indicating how certain the model is that the image belongs to a specific class.

Image classification is useful when you only need to know which objects are present in an image, without needing to determine their locations or shapes.

Figure 2.10: Image classification example

Object Detection

Object detection, on the other hand, involves both identifying the class and localizing the position of objects: it first classifies the detected objects into different categories based on their features and then draws bounding boxes around them.

Thus, the output of an object detector is a set of bounding boxes that enclose the objects in the image (Fig. 2.11), along with the corresponding class labels and confidence scores for each box.

Object detection is a good choice when it's necessary to identify objects of interest in a scene and determine their locations, but knowing their exact shapes is not essential.

Figure 2.11: Object detection example

Instance Segmentation

Instance segmentation takes object detection further by identifying individual objects in an image and accurately delineating their boundaries, separating each object from the background.

To achieve this, each pixel in the image is assigned a specific category label, so the output of an instance segmentation model includes a set of masks or contours

outlining each object in the image (Fig. 2.12), along with class labels and confidence scores for each object.

Instance segmentation is useful when you need to know not only where objects are in an image, but also their exact shapes.

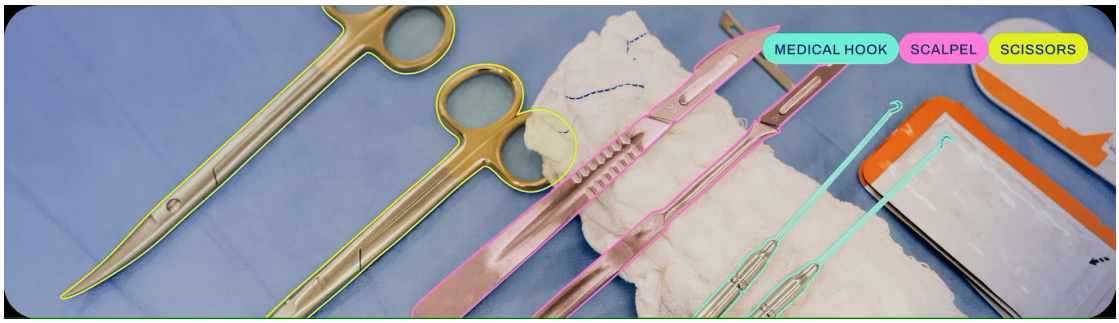

Figure 2.12: Instance segmentation example

Pose Estimation

Pose estimation is a task focused on detecting the locations of specific points, known as keypoints, which are used to track movements or poses.

These keypoints can represent various parts of an object, such as joints, landmarks, or other distinctive features, and are typically represented as a set of 2D or 3D coordinates.

The output of a pose estimation model is a set of points that mark the keypoints on an object in the image (Fig. 2.13), usually along with confidence scores for each point.

Pose estimation is ideal when you need to identify specific parts of an object in a scene and understand their relative positions to one another.

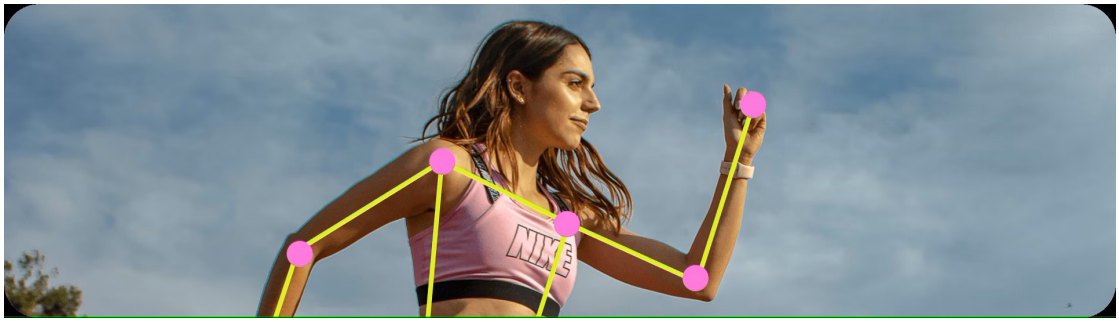

Figure 2.13: Pose estimation example

2.2.1 Ultralytics - YOLO (You Only Look Once)

Ultralytics [10] is a company specializing in AI and computer vision technologies, founded in 2019, that gained recognition through the development of open-source models and tools, making these technologies accessible to a wide range of users, from researchers to industrial developers, for real-world applications.

The company's namesake library (Fig. 2.14), built on PyTorch, includes tools to automatically train, test, and deploy models for various tasks on multiple devices.

Figure 2.14: Ultralytics logo

Ultralytics is renowned for developing optimized versions of YOLO, the acclaimed real-time object detection model originally created by Joseph Redmon and Ali Farhadi at the University of Washington in June 2015, as introduced in their research paper *"You Only Look Once: Unified, Real-Time Object Detection"* [11].

Prior approaches to object detection repurposed classifiers in a slow and complex pipeline that (1) generates potential bounding boxes, (2) runs a classifier on these proposed boxes and (3) post-process to refine the boxes, eliminate duplicate detections, and rescore them based on the presence of other objects in the scene.

In this way each component was trained separately, making the workflow heavy.

YOLO revolutionized this by framing object detection as a single regression problem, drawing inspiration from how humans can glance at an image and instantly recognize what objects are present, where they are, and how they interact.

It uses a simple pipeline (Fig. 2.15) that processes the image in one go to predict both the objects and their locations simultaneously.

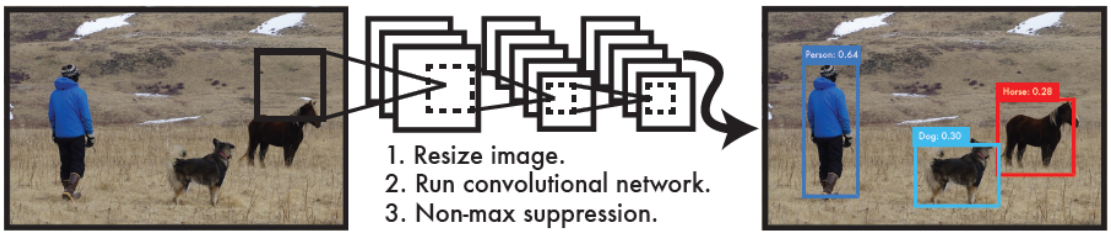

Figure 2.15: YOLO detection system. It (1) resizes the input image to 448x448, (2) runs a single CNN on it, and (3) thresholds the resulting detections by the model's confidence.

In particular, a single neural network predicts bounding boxes and class probabilities directly from the entire image, leveraging global context to make predictions across all classes simultaneously.

Unlike methods that rely on Region Proposal Networks, which detect potential regions of interest and then classify them in separate and multiple iterations, YOLO eliminates the need for this two-step process by using a single fully connected layer.

YOLO divides the input image into an SxS grid, where each grid cell is responsible for detecting objects whose centers fall within it.

Each grid cell predicts B bounding boxes and associated confidence scores, indicating both the likelihood that the box contains an object and the accuracy of the box's dimensions.

Each bounding box consists of five parameters:

- **x** and **y**: coordinates of the center of the box relative to the grid cell
- **w** and **h**: width and height of the box, relative to the entire image
- **confidence score**: indicating the certainty that the box contains an object and the accuracy of the bounding box

Since multiple bounding boxes may be generated for the same object, leading to overlapping or redundant predictions, YOLO uses Non-Maximal Suppression (NMS), a post-processing technique that removes unnecessary or incorrect bounding boxes (Fig. 2.16).

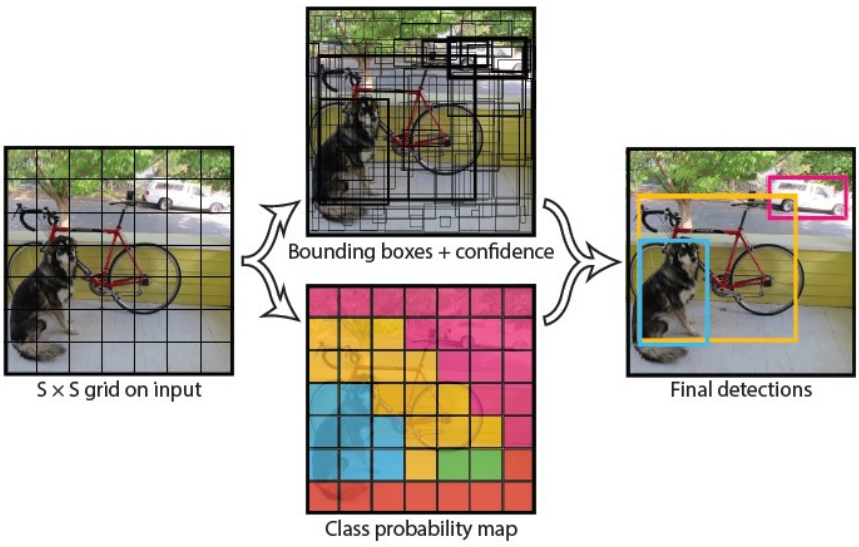

Figure 2.16: Non-Maximal Suppression (NMS) applied to a YOLO SxS grid

This step greatly enhances the clarity and accuracy of YOLO's object detection results by ensuring that only one bounding box per detected object is retained.

The architecture of the convolutional neural network (CNN) that serves as the backbone of YOLO is inspired by *GoogLeNet*, incorporating 24 convolutional layers to extract features from the image, followed by 2 fully connected layers to predict the output probabilities and coordinates (Fig. 2.17).

Instead of using *GoogLeNet*'s Inception modules, it employs a combination of 1x1 and 3x3 convolutional layers, to reduce the features space from preceding layers.

YOLO's convolutional layers are pretrained on the ImageNet classification task using a reduced half resolution of 224x224, that is then doubled for object detection, improving its accuracy without compromising speed.

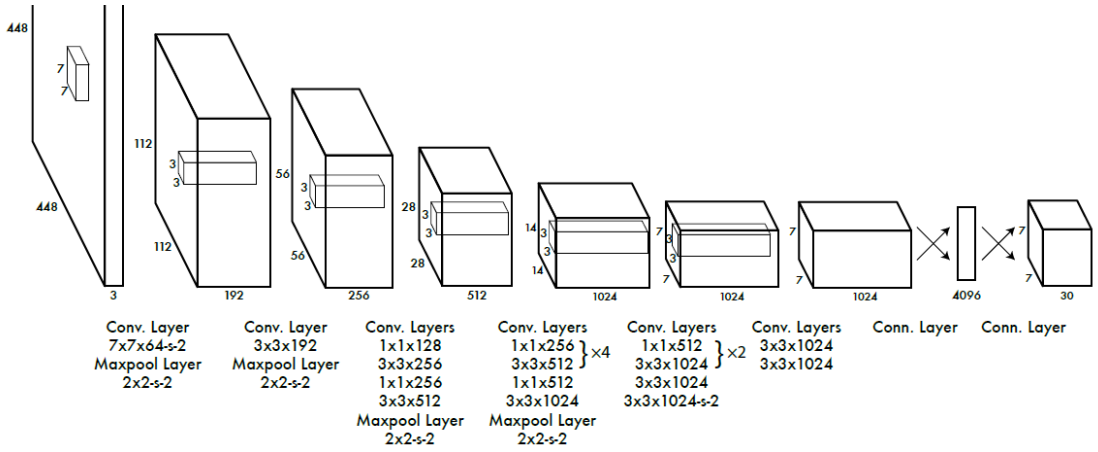

Figure 2.17: YOLO original architecture

Since its initial release in June 2015, several versions [12] (Fig. 2.18) of YOLO have introduced significant enhancements:

- YOLO v2 (Dec. 2016), also known as YOLO9000, introduced a different CNN backbone called Darknet-19 (a variant of the VGGNet architecture), batch normalization layer including a regularization to prevent overfitting, higher resolution of inputs (448x448), and convolution layers with anchor boxes.

- YOLO v3 (Mar. 2018) introduced Darknet-53 (a variant of the ResNet architecture), anchor boxes with different scales and aspect ratios, and feature pyramid networks to detect objects at multiple scales.

- YOLO v4 (Apr. 2020) introduced a new CNN architecture called CSPNet (still a variant of ResNet), dimensionality clustering to find anchors using K-Means, and Spatial Pyramid Pooling block to increase the receptive field.

- YOLO v5 (Jun. 2020) introduced EfficientDet (based on EfficientNet), and a new method for generating the anchor boxes called "dynamic anchor boxes".
- YOLO v6 (Jun. 2022) introduced a variant of the EfficientNet architecture called EfficientNet-L2, and a new method for generating the anchor boxes called "dense anchor boxes".
- YOLO v7 (Jul. 2022) introduced the Extended Efficient Layer Aggregation Network (E-ELAN), the processing of images at a resolution of 608x608 pixels, and a concept known as "trainable bag-of-freebies".
- YOLO v8 (Jan. 2023) introduced anchor-free split Ultralytics head, and advanced CSPDarkNet backbone and path aggregation.
- YOLO v9 (Feb. 2024) introduced a new technique that optimizes gradient flow during training called “Programmable Gradient Information” (PGI), and an enhancement that further improves feature learning and aggregation called “Generalized Efficient Layer Aggregation Network” (GELAN).
- YOLO v10 (May 2024) introduced an enhanced version of CSPNet (Cross Stage Partial Network) as backbone, the elimination of non-maximum suppression (NMS) thanks to the One-to-One Head that generates a single best prediction per object, and Path Aggregation Network layers for multiscale feature fusion.
- YOLO 11 (Sep. 2024) introduced an improved backbone with C3K2 blocks and neck architecture, SPFF (Spatial Pyramid Pooling Fast), and advanced attention mechanisms like C2PSA.

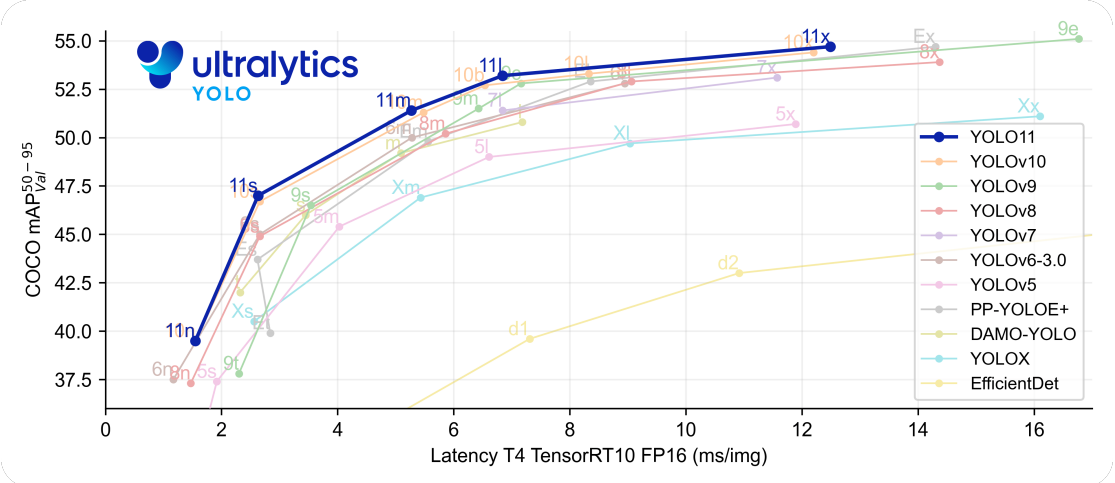

Figure 2.18: YOLO models performance comparison

2.2.2 Ultralytics - SAM (Segment Anything Model)

Ultralytics has also integrated SAM by Meta AI [13] (Apr. 2023), a groundbreaking model for zero-shot segmentation capable of segmenting any object in an image without requiring specific training for that object, thanks to his ability to learn a general concept of what constitutes an "object".

The goal of SAM is to develop a foundation model for many segmentation tasks, inspired by the success of foundation models in NLP that exhibit strong zero-shot and few-shot generalization using prompting techniques, carefully crafted text inputs that guide the model to generate meaningful outputs.

Similarly, SAM leverages flexible prompts to guide its segmentation process, introducing a promptable interface (Fig. 2.19) that enables users to specify what to segment in an image through various forms of input, such as points, bounding boxes, textual prompts, or edges.

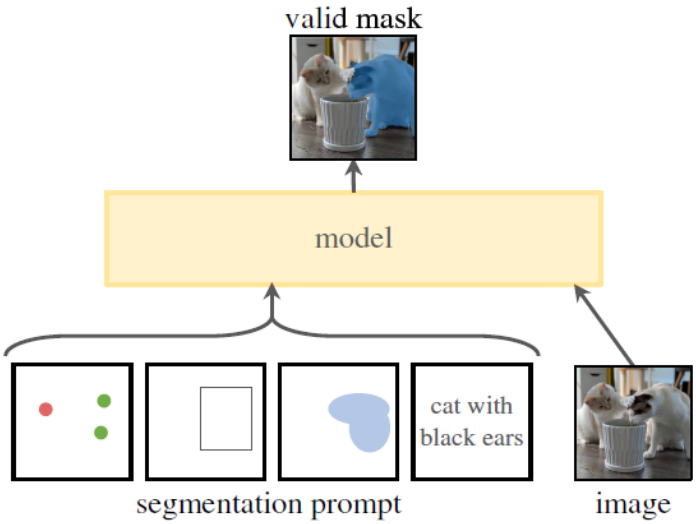

Figure 2.19: SAM flexible promptable interface

The output is a *valid* mask that delineates the selected object in real time, where the "valid" requirement ensures that even when a prompt is ambiguous (e.g., overlapping objects or unclear references), the output is reasonable and corresponds to at least one of the intended objects.

The architecture of SAM (Fig. 2.20) is built on three components:

- an Image Encoder that processes the input image to compute an high dimensional image embedding, capturing rich visual features
- a Prompt Encoder that converts user-provided prompts into embeddings

- a Mask Decoder that combines the image and prompt embeddings to predict segmentation masks

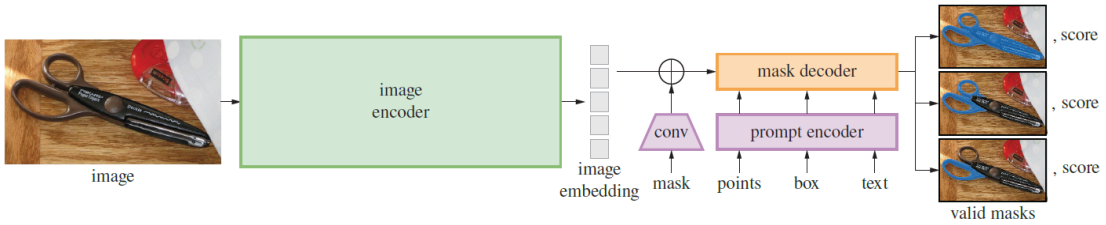

Figure 2.20: SAM architecture

By separating SAM into a powerful image encoder and a lightweight prompt encoder/mask decoder, the same image embedding can be reused with different prompts and its cost amortized.

SAM 2 [14] (Oct. 2024) is the extension of SAM to video (Fig. 2.21), by considering images as a video with a single frame, whose design is a simple transformer architecture equipped with a memory attention module, that stores information about the object and previous interactions.

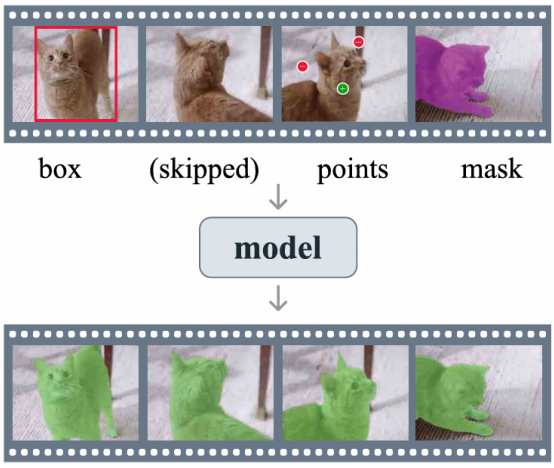

Figure 2.21: SAM 2 promptable video segmentation

This allows it to generate mask predictions throughout the video and effectively correct them based on the stored memory context of the object from previously observed frames.

In detail, the components (Fig. 2.22) in SAM 2 are:

- a hierarchical Image Encoder (allowing the use of multiscale features during decoding), which is run only once for the entire interaction to provide

unconditioned tokens (feature embeddings) representing each frame

- a Memory Attention module, which conditions the current frame features on the past frame features and predictions as well as on any new prompts
- a Prompt Encoder, identical to the one in SAM, which can be prompted by clicks (positive or negative), boxes, or masks to define the extent of the object in a given frame
- a Mask Decoder, whose design largely follows SAM's but differs in how it ensures valid masks. Unlike SAM, where there is always a valid object to segment given a positive prompt, in video (where ambiguity can extend across frames), it is possible for no valid object to exist in some frames, for example due to occlusion
- a Memory Encoder, which generates memory of frames by downsampling the output mask using a convolutional module and then summing it element-wise with the unconditioned frame embedding from the Image Encoder, placing them in a bank
- a Memory Bank, which retains information about past predictions for the target object in the video by maintaining a FIFO queue of memories of up to N recent frames and storing information from prompts in a FIFO queue of up to M prompted frames

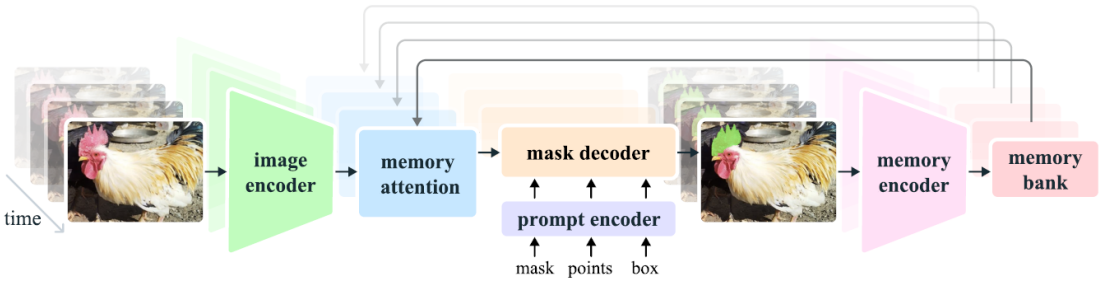

Figure 2.22: SAM 2 architecture

It is important to know that the frame embedding used by the SAM 2 decoder is not directly from an image encoder, but is instead conditioned on memories of past predictions and prompted frames, which can also come "from the future" relative to the current frame.

2.3 Generative AI

Generative Artificial Intelligence (GenAI) is a subset of artificial intelligence that employs generative models to create text, images, or other types of data [15].

These models analyze the underlying patterns and structures in their training data, enabling them to generate new data based on the given input, which is often provided in the form of natural language prompts.

Generative AI is applied across various industries, including software development, healthcare, finance, entertainment, customer service, sales and marketing, art, writing, fashion, and product design.

However, concerns have been raised about its potential misuse, such as enabling cybercrime, spreading fake news or deepfakes to deceive or manipulate people, and leading to the large-scale replacement of human jobs.

Additionally, intellectual property law issues arise with generative models trained on and mimicking copyrighted works of art (Fig. 2.23).

Figure 2.23: "Théâtre D'opéra Spatial", an image made using Generative AI

From Discriminative models...

Since its inception, the field of machine learning has employed both discriminative and generative models to analyze and predict data: these approaches are fundamentally different, making each suited to specific tasks.

In general, discriminative models focus on dividing the data space into classes

by learning the boundaries between them, whereas generative models aim to understand how the data are embedded within the space (Fig. 2.24).

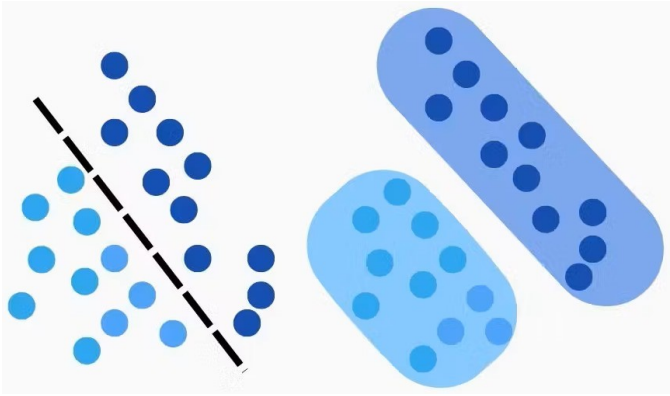

Figure 2.24: Discriminative (left) and Generative (right) models difference

Starting in the late 2000s, the rise of deep learning significantly advanced research and performance in tasks such as image classification, speech recognition, and natural language processing.

During this period, neural networks were predominantly trained as discriminative models, largely due to the challenges associated with generative modeling.

...to the first Generative models

In 2014, significant advancements led to the development of the first practical deep neural networks capable of learning generative models, including:

- Variational Autoencoders (VAEs), models that generate new data as variations of the input data they are trained on.

  Unlike traditional autoencoders, which encode a discrete, fixed representation of latent variables, VAEs encode a continuous, probabilistic representation of the latent space: this enables them not only to reconstruct the original input accurately, but also to use variational inference to generate new data samples that resemble the input data

- Generative Adversarial Networks (GANs), that consist of two neural networks working together, a generator that creates image samples that resemble the training dataset, and a discriminator that, on the other hand, evaluates whether a given image is a "real" image from the training data or a "fake" image generated by the generator.

These networks are trained jointly through a zero-sum game (Fig. 2.25), where feedback from the discriminator helps the generator improve its outputs, until the discriminator can no longer distinguish between real and fake samples.

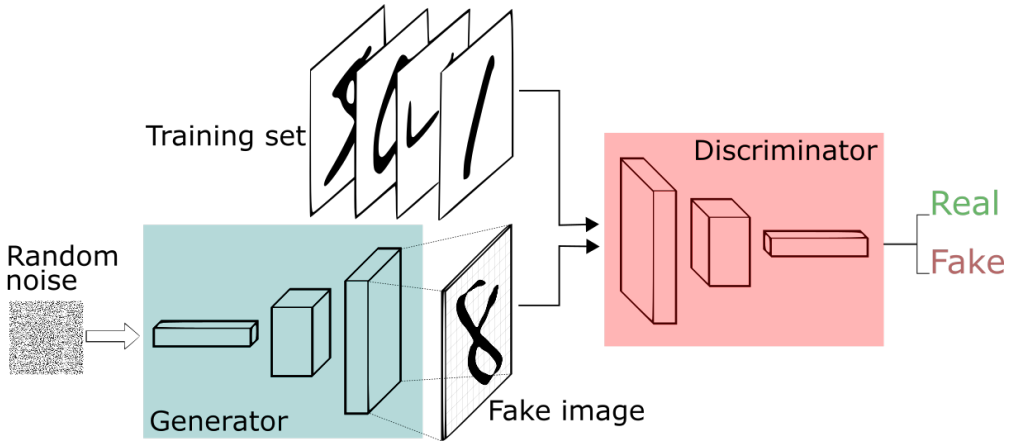

Figure 2.25: Generative Adversarial Networks (GANs) workflow example

These deep generative models were the first to output not only class labels for images but also entire images.

Long Short-Term Memory (LSTM)

Following these advancements came the era of Long Short-Term Memory (LSTM) [16], a type of recurrent neural network (RNN) designed to address the vanishing gradient problem that traditional RNNs (Fig. 2.26) often face during back-propagation.

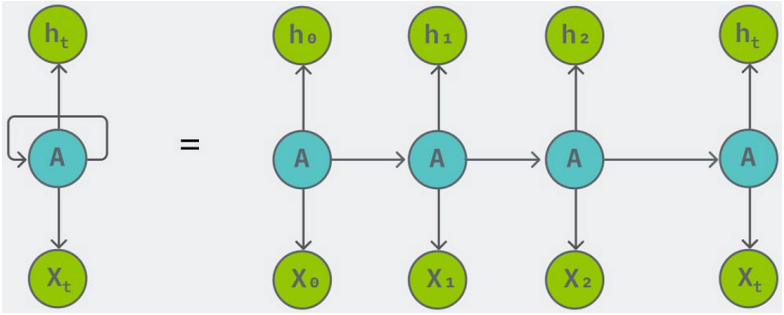

Figure 2.26: Recurrent Neural Network (RNN) standard architecture

LSTMs are distinguished by their ability to maintain information over extended sequences, by providing a short-term memory mechanism that can persist for thousands of timesteps.

An LSTM unit (Fig. 2.27) typically consists of a cell, which retains values over arbitrary time intervals, and three gates that regulate the flow of information into and out of the cell:

- Forget gate, determining which information from the previous state should be discarded. It maps the previous state and the current input to a value between 0 and 1, where a value of 1 indicates retention of the information and a value of 0 indicates discarding it.
- Input gate, deciding which pieces of new information should be stored in the current cell state, using a mechanism similar to the forget gate.
- Output gate, controlling which pieces of information from the current cell state should be output, assigning a value to the information based on the previous and current states.

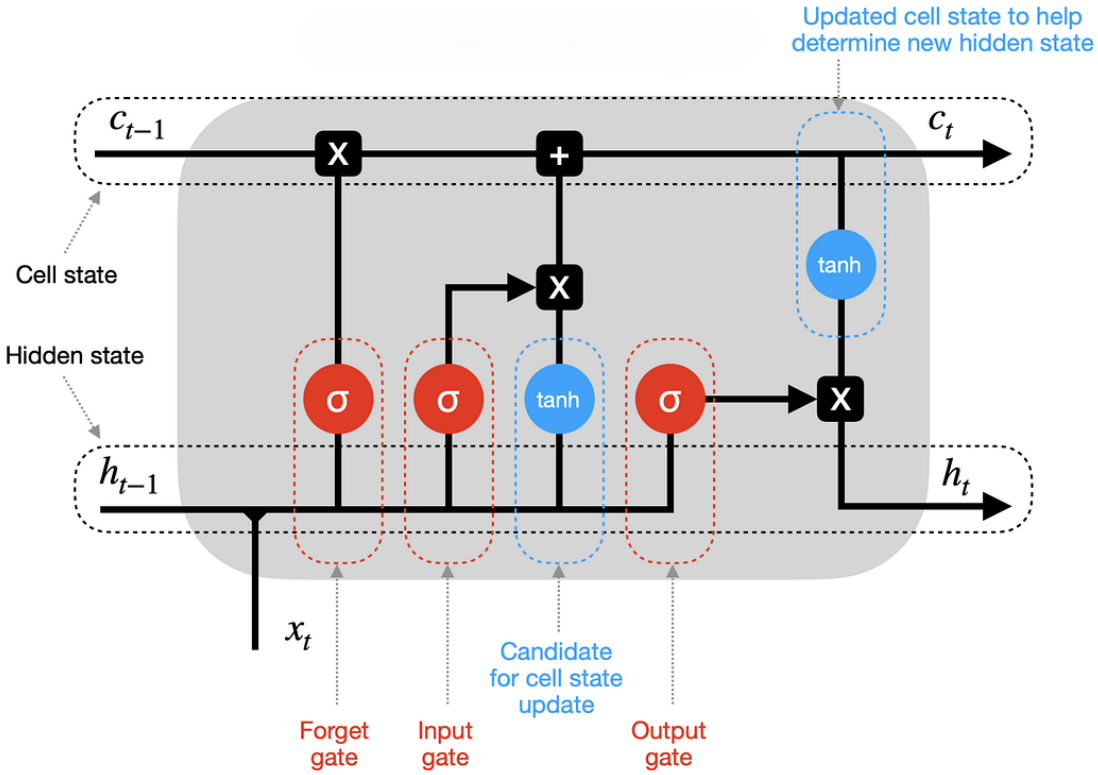

Figure 2.27: Long short-term memory (LSTM) recurrent unit

By selectively outputting relevant information from the current state, LSTMs can maintain useful long-term dependencies, enabling them to make predictions across both current and future timesteps.

However, LSTM networks are not without limitations, as they can still encounter the exploding gradient problem and are constrained by their sequential processing nature, as they process tokens one at a time, from the first to the last, like most other RNNs: this limitation prevents them from operating in parallel over all tokens in a sequence.

Sequence-to-Sequence (Seq2Seq)

LSTM remained the standard architecture for long sequence modeling until the publication of the 2017 paper "*Attention Is All You Need*" [17], which described sequence-to-sequence models and introduced the Transformer architecture.

Sequence-to-Sequence (or Seq2Seq) [18] refers to a deep learning architecture that transforms an input sequence, like a sequence of words in a sentence, into a different sequence.

Seq2Seq models are particularly effective in tasks like language translation (Fig. 2.28), where a sequence of words in one language is converted into a sequence of words in another language.

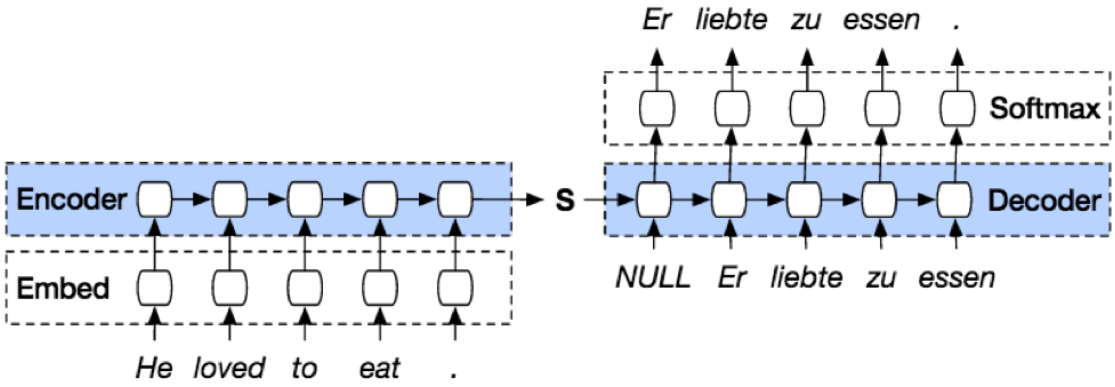

Figure 2.28: Sequence-to-Sequence (Seq2Seq) model in language translation

A common choice for Seq2Seq models has been Long Short-Term Memory (LSTM)-based models; this is because LSTM modules are well-suited for sequence-dependent data (for example sentences, where the order of words is crucial for understanding), for which they can interpret the sequence while selectively remembering important parts and forgetting unimportant ones.

A Seq2Seq model typically consists of:

- an Encoder, that processes the input sequence and maps it into a higher-dimensional space, represented as an n-dimensional vector
- a Decoder, which takes the abstract vector produced by the Encoder and transforms it into an output sequence

As mentioned, a simple implementation of a Seq2Seq model can use a single LSTM for both the Encoder and the Decoder.

Transformers

To better understand Transformers, it is essential to first grasp the concept of the attention mechanism, which examines an input sequence and determines, at each step, which other parts of the sequence are important.

This might sound abstract, but it is similar to the process we use when reading a text: while focusing on the word we are currently reading, our mind simultaneously retains key keywords from the text to provide context.

In technical terms, for each input processed by the Encoder, the attention mechanism evaluates other inputs in the sequence and assigns different weights to them, based on their importance: this allows the model to focus on the most relevant parts of the sequence when encoding information.

The Decoder then uses the encoded representation of the sequence, along with the weights assigned by the attention mechanism (Fig. 2.29), to generate the output sequence.

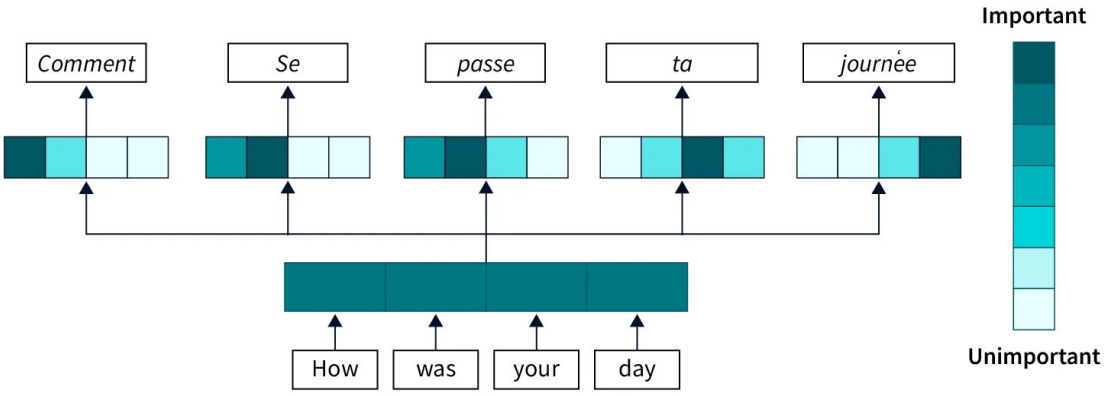

Figure 2.29: Example of the "Attention" mechanism

As anticipated, the landmark paper "*Attention Is All You Need*" presented at NeurIPS conference by Google researchers also introduced a novel architecture called Transformer, that uses the attention-mechanism we saw earlier and, like LSTM-based models, transforms one sequence into another one with the help of two parts, the Encoder and the Decoder.

However, the Transformer [19] (Fig. 2.30) differs from the previously described or existing sequence-to-sequence models, because it has the advantage of having no recurrent units, therefore requiring less training time compared to earlier recurrent neural network (RNN) architectures.

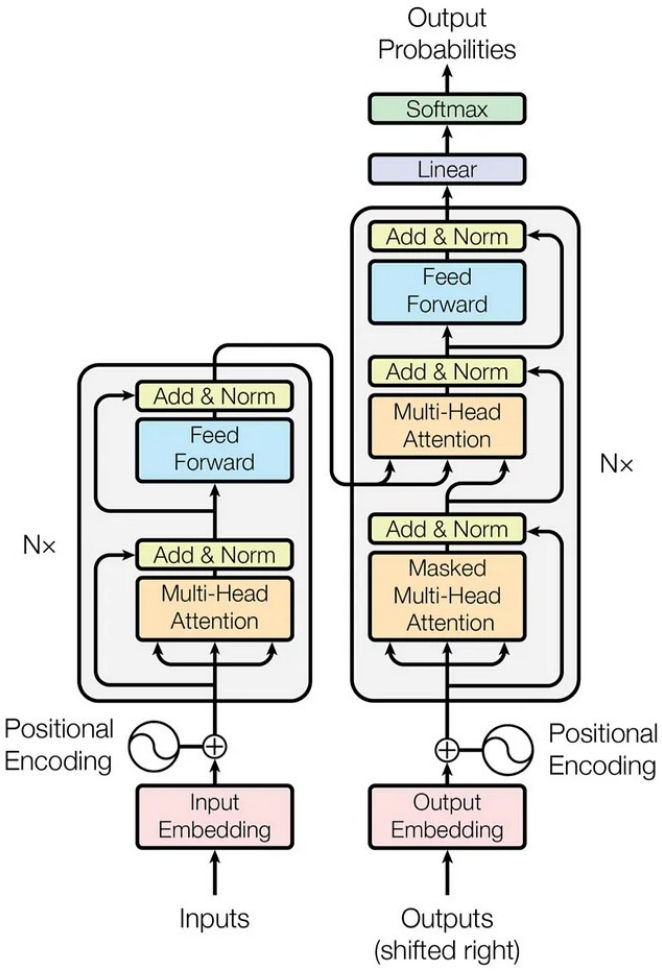

Figure 2.30: Transformer architecture

Both the Encoder and Decoder consist of modular components that can be stacked multiple times, as represented by "Nx" in the figure: these components mainly include Multi-Head Attention and Feed Forward layers.

Since raw text cannot be processed directly, inputs and outputs are first embedded into an n-dimensional space.

A key aspect of the model is positional encoding, which assigns a distinct position to each word in the sequence; that is because, unlike recurrent neural networks which inherently preserve input order, the Transformer requires positional encoding to capture the sequence structure. These positional values are added to the embedded representation (n-dimensional vector) of each word to encode the relative order of words in the sequence.

Another fundamental element is the Multi-Head Attention layer, which we will

analyze in detail by first examining the Attention mechanism, that can be described by the following equation:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

in which:

- Q is the matrix of the query (vector representation of one word in the sequence)
- K are all the keys (vector representations of all the words in the sequence)
- V are the values, i.e. the vector representations of all the words in the sequence

To simplify, we could say that the values in V are multiplied and summed with some attention-weights a, defined by:

$$a = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$$

This implies that the weights a are determined by how each word in the sequence (represented by Q) is influenced by all other words (represented by K).

To ensure a normal distribution, the SoftMax function is applied such that the resulting values are then used to weight all words in the sequence, which are introduced in V (where Q and V are the same for the encoder and decoder but differ in the module that connects them).

This attention mechanism can be parallelized into multiple instances operating simultaneously (Fig. 2.31).

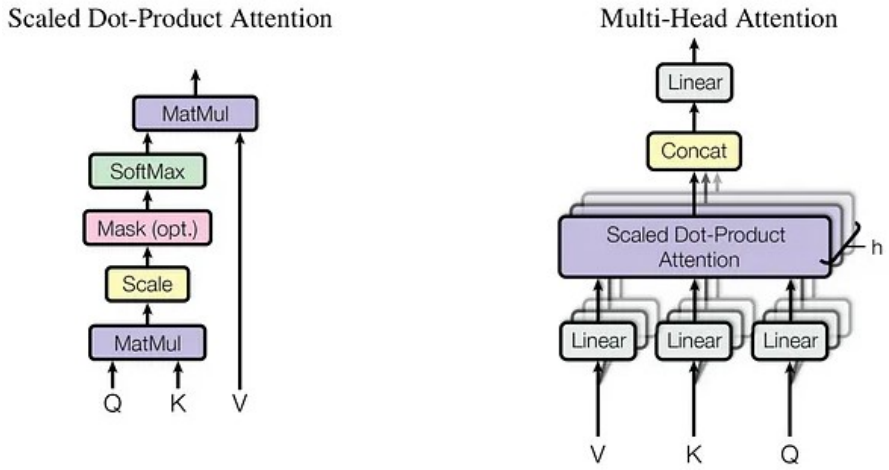

Figure 2.31: Attention and Multi-Head Attention layers

By doing so, the attention process is repeated several times, each using different linear projections of Q, K, and V.

This enables the system to learn from different representations of Q, K, and V by applying weight matrices W, which are learned during training: this flexibility enhances the model's ability to capture complex patterns.

The matrices Q, K, and V vary depending on their position within the attention modules—whether in the encoder, decoder, or the connection between them.

This distinction is necessary because the attention mechanism needs to focus either on the entire encoder input sequence or only a portion of the decoder input sequence.

The multi-head attention module linking the encoder and decoder ensures that both the encoder's input sequence and the decoder's input sequence (up to a certain position) are considered together.

Following the multi-head attention layers in both the encoder and decoder, a pointwise feed-forward layer is applied.

This feed-forward network is the same for every position in the sequence and can be seen as applying an identical linear transformation independently to each element in the sequence.

Embeddings from Language Model (ELMo)

Transformers were initially developed as an improvement over earlier deep learning architectures for machine translation, but they became the foundation for an AI boom in generative systems during the early 2020s: to make the discussion simpler and more understandable, let's break it down step by step.

An instrumental step in the evolution toward transformer-based language modeling was ELMo (Embeddings from Language Model) [20], a word embedding method that represents a sequence of words as a corresponding sequence of vectors.

ELMo was historically significant as a pioneer of self-supervised generative pretraining followed by fine-tuning: in this approach, a large model is first trained on a large corpus to reproduce text, and then augmented with additional task-specific weights and fine-tuned on supervised task data for specific applications.

After the 2017 publication of Transformer architecture, the architecture of ELMo was changed from a multilayered bidirectional LSTM to a Transformer encoder, giving rise to BERT.

Bidirectional Encoder Representations from Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) [21] is a language model introduced in October 2018 by Google researchers, that can be considered as a first example of what today a Large Language Model (LLM) is.

BERT is trained by masked token prediction and next sentence prediction: this way it learns contextual, latent representations of tokens in their context, using self-supervised learning.

Although the original Transformer architecture includes both encoder and decoder, BERT is an encoder-only model (Fig. 2.32) comprising four main modules:

- Tokenizer, a module that converts a piece of text into a sequence of integer "tokens"
- Embedding, that transforms the sequence of discrete tokens into an array of real-valued vectors, mapping into a lower-dimensional Euclidean space
- Encoder, a stack of multiple transformer blocks with self-attention mechanisms, but without causal masking
- Task head, a module that converts the final vector representations into one-hot encoded tokens again, by producing a predicted probability distribution over the token types; it serves as a simple decoder, transforming latent representations back into token types, and is sometimes referred to as an "un-embedding layer"

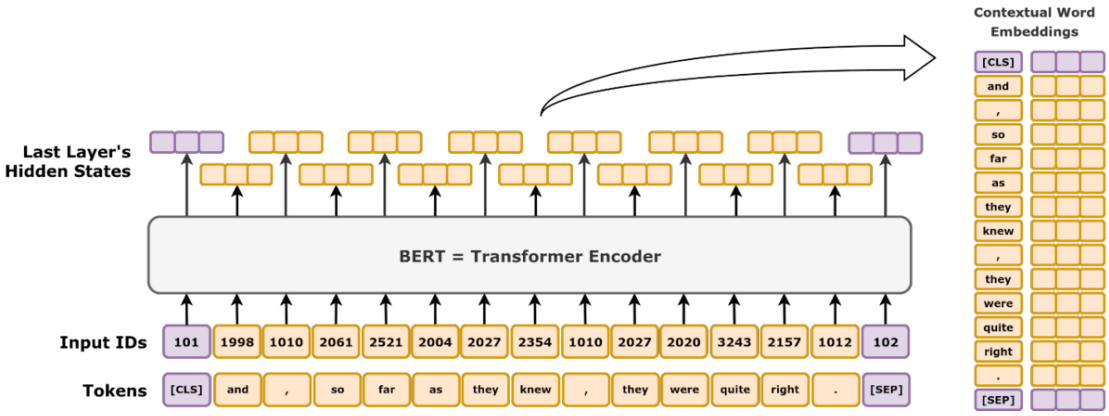

Figure 2.32: Bidirectional Encoder Representations from Transformers (BERT) model

By 2023, academic and research usage of BERT began to decline due to rapid advancements in decoder-only models (on which are based LLMs), which demonstrated simplicity, good zero-shot generalization, and cheaper training cost to attain a reasonable performance.

Large Language Models (LLMs)

Large Language Models (LLMs) are a class of advanced foundation models trained on vast amounts of textual data; these models are capable of understanding and generating natural language, enabling them to support numerous use cases, applications, and a wide range of tasks.

As language models, LLMs acquire these abilities by learning statistical relationships from extensive text corpora, during a combination of self-supervised and semi-supervised training processes.

In general, an LLM operates by being provided with a "prompt", a combination of input data and instructions specifying what to do with that input, so that, based on the prompt, the LLM generates an output tailored to the particular use case described (Fig. 2.33).

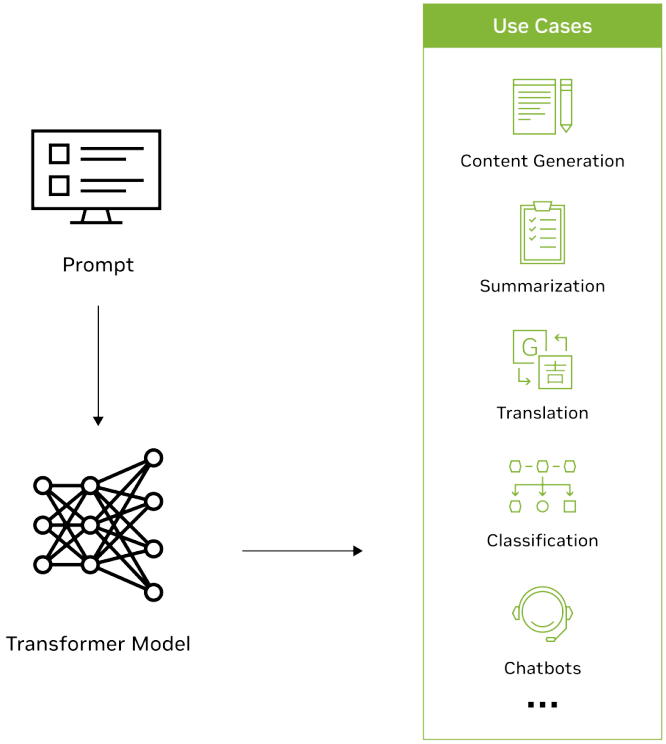

Figure 2.33: Example on how an LLM works

Their versatility contrasts sharply with the traditional method of building and training domain-specific models for individual tasks: so it is easily understandable why, due to their size and computational demands, LLMs are typically too large to run on a single computer. Instead, they are offered as a service through APIs or web interfaces, eliminating the need for local downloads or installations.

Multimodal Large Language Models (LLMs)

But while LLMs have demonstrated remarkable capabilities in understanding and generating text, there are many scenarios where we require them to work with more than just textual data.

For instance, we might want them to handle multiple modalities of data, where a modality refers to a specific type or form of information, such as text, images, audio, or video.

To address this need, Multimodal LLMs [22] (Fig. 2.34) were developed: these models extend traditional LLMs by incorporating multimodality, meaning they can process and generate content across several data types, enhancing even more their versatility and broadening their understanding of real-world phenomena.

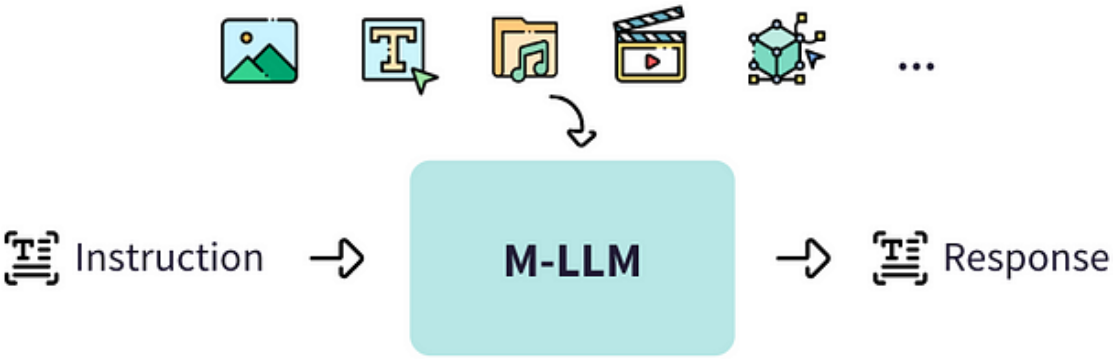

Figure 2.34: Example on how a Multimodal LLM works

When dealing with multimodal data, it is crucial to use a model that can jointly represent information from different modalities: this enables the model to capture and leverage the combined insights offered by these diverse forms of data.

A common approach to creating multimodal models from an LLM involves tokenizing the output of a trained encoder (a vision encoder is used to process images or video, while an audio encoder is employed to handle audio files).

Using this method, a wide range of multimodal LLMs have been developed, including Audio-Text-to-Text, Image-Text-to-Text, Visual Question Answering, Document Question Answering, Video-Text-to-Text, and more in general Any-to-Any, opening up virtually infinite possibilities for use cases and applications.

2.3.1 OpenAI - Generative Pre-trained Transformer (GPT)

Among the various LLM providers currently on the market, the most well-known is undoubtedly OpenAI (Fig. 2.35), an American AI research organization founded in December 2015 and headquartered in San Francisco, California.

Figure 2.35: OpenAI logo

OpenAI's stated mission is to develop "safe and beneficial" Artificial General Intelligence (AGI), which it defines as "highly autonomous systems that outperform humans at most economically valuable work" [23].

The organization comprises the non-profit OpenAI, Inc., and its for-profit subsidiary introduced in 2019, OpenAI Global, LLC, on which Microsoft holds approximately 49% equity, having invested $13 billion, while also providing computing resources through its cloud platform, Microsoft Azure.

OpenAI is primarily known for ChatGPT, an AI tool launched in November 2022, built on the proprietary Generative Pre-trained Transformer (GPT) [24] family of LLMs and fine-tuned for conversational applications through a combination of supervised learning and reinforcement learning from human feedback (RLHF).

ChatGPT enables users to ask questions in natural language (Fig. 2.36), responding within seconds while considering previous user prompts and replies as context, and moreover delivering answers tailored to the desired length, format, style, level of detail, and language.

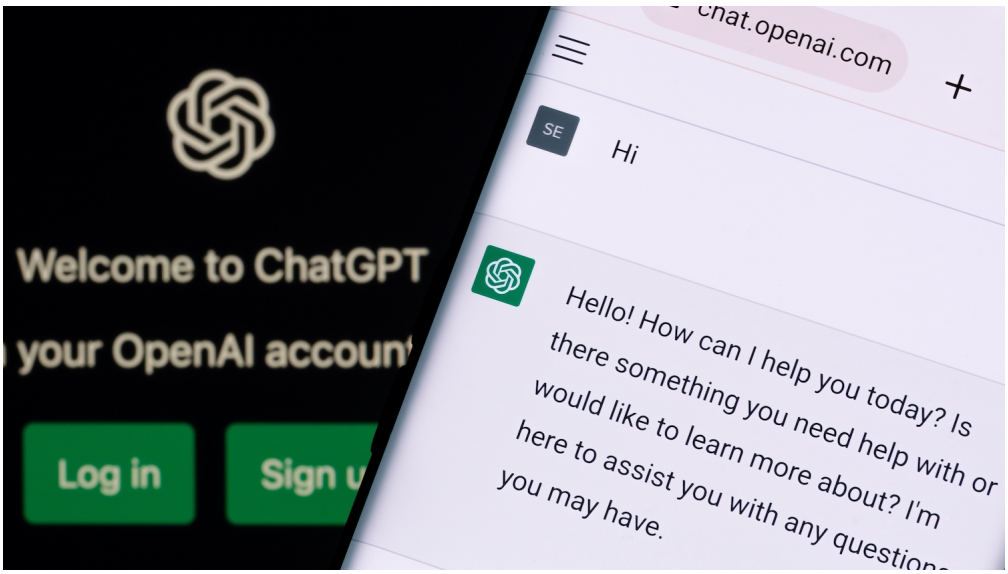

Figure 2.36: ChatGPT web interface

The release of ChatGPT marked a significant milestone in the AI boom as, by

January 2023, it had become the fastest-growing consumer software application in history, reaching over 100 million users within two months.

Let's revisit the timeline of releases for the various models in the GPT family:

- In June 2018 the original paper on the generative pre-training of a transformer-based language model, referred to as "GPT-1", was published as a preprint on OpenAI's website.

  It demonstrated how a generative language model could acquire world knowledge and process long-range dependencies by pre-training on a diverse corpus containing long stretches of contiguous text.

- In February 2019, the same year OpenAI transitioned from a non-profit to a "capped" for-profit organization, the company announced GPT-2, an unsupervised transformer language model that gained attention for its ability to generate human-like text, and served as the successor to OpenAI's original GPT model.

- In May 2020, OpenAI introduced GPT-3, a language model trained on extensive internet datasets and designed for tasks such as answering natural language questions, translating between languages, and generating coherent, improvised text.

  OpenAI stated that the full version of GPT-3 contained 175 billion parameters, two orders of magnitude larger than the 1.5 billion parameters in GPT-2: this dramatic increase significantly improved benchmark results over GPT-2.

- In November 2022, OpenAI launched a free preview of ChatGPT, an AI chatbot based on their new GPT-3.5.

- In March 2023, OpenAI announced GPT-4, capable of processing both text and image inputs and writing code in all major programming languages.

  From this point onward, OpenAI declined to disclose various technical details and statistics about GPT models, such as their precise size.

- In May 2024, OpenAI announced and released GPT-4o ("o" for "omni"), which can process and generate text, images, and audio, achieving state-of-the-art results in voice, multilingual, and vision benchmarks.

- In July 2024, OpenAI introduced GPT-4o mini, a smaller and more cost-effective version of GPT-4o, priced at $0.15 per million input tokens and $0.60 per million output tokens (compared to $5 and $15, respectively, for GPT-4o), designed for enterprises, startups, and developers seeking to automate services with AI agents.

- In September 2024, OpenAI released the o1 (or Strawberry) and o1-mini models, which have been designed to take more time to think, generating a long "chains of thought" before returning a final answer: this approach improved performance in tasks requiring scientific reasoning, coding, and complex problem-solving.

  OpenAI described o1 as a complement to GPT-4o rather than a successor, as this ability to think before responding represents a new, additional paradigm, which is improving model outputs by spending more computing power during the generation process (Fig. 2.37), in contrast to the classic scaling paradigm, enhancing outputs by increasing model size, training data and computational resources.

Figure 2.37: OpenAI o1 (or Strawberry) new scaling paradigm

- In December 2024 OpenAI unveiled o3, the successor to the o1 reasoning model, along with its lighter and faster version o3-mini, further expanding the range of available models.

Chapter 3

Proposed tasks

3.1 Reading of gauge measurements

Context and motivation

A gauge is an instrument available in various forms and materials (Fig. 3.1), designed to measure pressure (but not only) from gases, liquids, vapors, or solid bodies across numerous applications and industries.

Figure 3.1: An industrial machinery equipped with digital and analog gauges

They are critical components in most industrial processing systems, where they must be reliable, accurate, and easy to read to ensure safety and maintain workflow integrity in daily operations: for instance, detecting an overheating motor in advance can make the difference between a minor repair and a complete replacement [25].

However, it is not just about preventing operational downtime, which can be costly; equipment failures can also be dangerous and, in certain circumstances,

potentially life-threatening.

Consider a reactor (Fig. 3.2) where chemical reactions generate heat: the temperature and pressure gauges in the reactor can reveal a thermal runaway or the accumulation of excess energy, where chemicals react more rapidly, escalating the risk of an explosion.

Figure 3.2: A chemical Continuous Stirred Tank Reactor (CSTR)

In such situations, safety mechanisms like a shutoff valve, the introduction of inert elements, or rapid cooling can prevent disaster; however, before executing these methods, it is crucial to recognize the need for action.

Thus, the first step is to establish a system capable of reading gauge data, processing the information, and determining whether measurements have reached a critical threshold: in cases where thresholds are breached, it is fundamental to respond immediately.

Setup

To monitor the gauge measurements in industrial plants, it is essential to first consider the specific use case: is continuous monitoring required, or is it sufficient to check the gauge at fixed intervals?

In the first case, the issue can be addressed using a classic surveillance camera directed at the pressure gauge while, in the second case, it is much more efficient to leverage what we previously introduced as Autonomous Mobile Robots (AMRs).

As mentioned earlier, these robots can be configured to perform autonomous "missions" (Fig. 3.3), namely patrol routes with predefined stops at strategic points

and frequencies, during which they collect a variety of data in multiple formats using the cameras and sensors with which they are equipped.

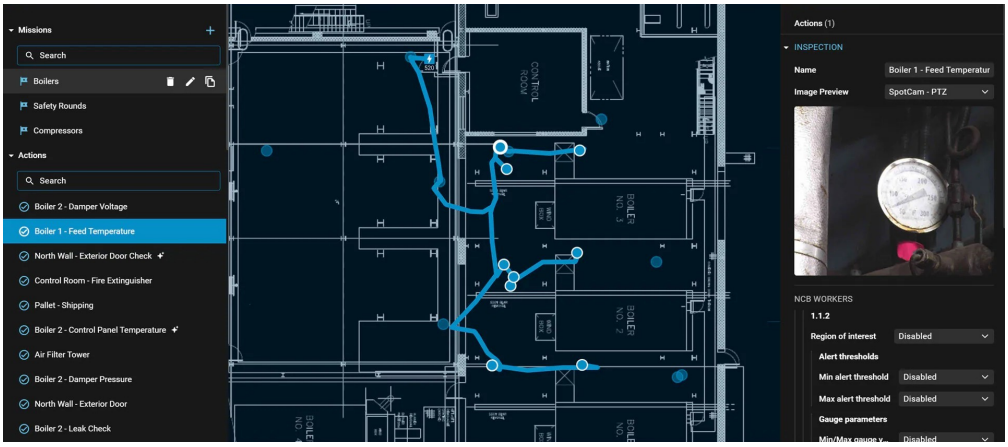

Figure 3.3: An example of autonomous mission configuration

In the specific use case I analyzed, continuous monitoring was not necessary; however, the site in question was a large facility with a vast number of machines equipped with pressure gauges.

Therefore, the most functional choice was Spot, which, as described earlier, can be considered an IoT platform with customizable accessories.

For this purpose, I chose to equip Spot with:

- a Pan-Tilt-Zoom (PTZ) camera with 30x optical zoom and a 360° x 170° field of view (Fig. 3.4, on the head), to allow readings even at significant distances.

Figure 3.4: Spot equipped with payloads

- a LIDAR (Light Detection And Ranging) sensor (Fig. 3.4, on the back), a tool that determines the distance of an object surface by measuring the time between the emission of a laser pulse and the reception of the reflected signal.

  Integrated with Spot's Reinforcement Learning techniques, this sensor helps the robot navigate a constantly changing environment, such as an industrial plant: in fact, using the LIDAR sensor, Spot can map its surroundings up to a distance of 100 meters (Fig. 3.5) and, if a large obstacle blocks its path, the system calculates the shortest alternative route to reach the destination.

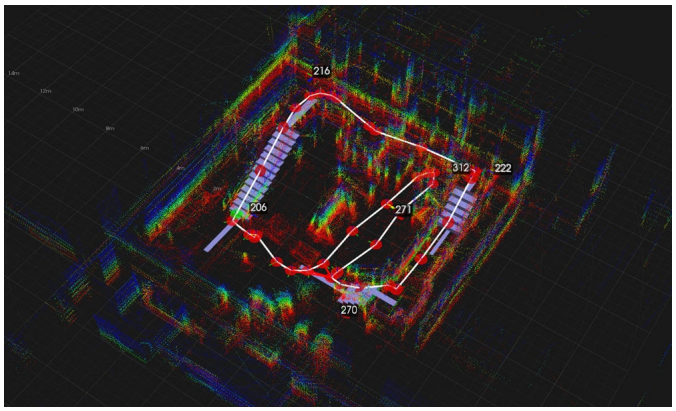

Figure 3.5: Spot's LIDAR mapping

- a Dock recharging station (Fig. 3.6) that automatically supplies power to recharge Spot's batteries via a dedicated connection on the rear alignment tower: this allows Spot to autonomously return for recharging when necessary.

Figure 3.6: Spot's docking station

Charging is automatically initiated by Spot after it detects the fiducial sticker (similar to a QR code), properly aligns itself on the dock, and activates the power flow by sitting on top of it.

The final pipeline ensures that Spot performs its inspection rounds at regular, adjustable intervals, capturing photos (Fig. 3.7) of all pressure gauges included in its mission before returning to its docking station until the next iteration.

Figure 3.7: Spot's autonomous gauge mission

Simultaneously, every photo is immediately sent for processing to a system that reads the measurement displayed on the gauge and acts based on where the value falls within the optimal or allowed range: for example, it may request further verification by human personnel (Fig. 3.8) or directly trigger an alert.

Figure 3.8: Verification by human personnel of an anomalous measurement

However, not all pressure gauges can be processed in the same way, as they are divided into digital and analog types: these differences will be analyzed in detail in the following chapters.

3.1.1 Digital gauges

From OCR to Multimodal Large Language Models

In general, a digital gauge is a device equipped with a display that shows one or more values (Fig. 3.9), each associated with its corresponding unit of measurement.

Figure 3.9: An example of digital gauge

A common approach (the one used until now) involves the use of Optical Character Recognition (OCR) [26], an AI field of research within computer vision and image pattern recognition, which is the electronic or mechanical conversion of images of handwritten or printed text into machine-encoded text (Fig. 3.10).

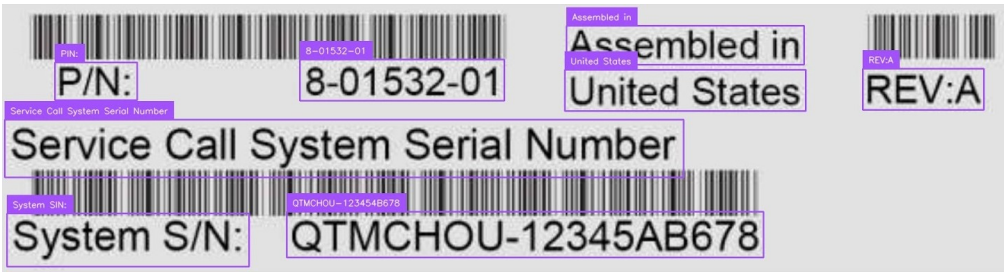

Figure 3.10: Optical Character Recognition (OCR) of a label

Early versions of OCR required training with images of each character and were

limited to working with one font at a time; instead, advanced systems capable of achieving high accuracy with most fonts are now common, along with support for a variety of image file formats.

However, even though OCR models are capable of structuring their output in formats like json, it is not possible to provide them with instructions so they are not able to filter which content to keep and which one not to, unless a support script is provided.

For this reason, what I was specifically requested for was a focus on the OCR capabilities integrated into modern Multimodal Large Language Models (LLMs), which are more flexible in handling different types of input data or recognizing various kinds of text.

To this end, I decided to compare various models, some available as services through API calls, as they are too large and computationally intensive to run locally; others, due to their smaller size, can be executed on-premises (on-prem, as opposed to software as a service, or SaaS).

Among the first category, I selected OpenAI's GPT-4, GPT-4o, and GPT-4o mini, the latest released models, excluding o1 and o1-mini, which, however, do not promise noteworthy improvements for an OCR task.

As for the second group, it is necessary to first make a distinction based on the hardware capabilities available to support the chosen models.

Indeed, LLMs are computationally expensive to run, as they require substantial memory to store the model parameters and intermediate calculations at inference time.

System memory (RAM) is not ideal for this purpose because it is slower than GPU memory, specifically designed for high-performance computing tasks such as deep learning, offering the speed and bandwidth required to efficiently perform the complex computations of LLMs, without encountering bottlenecks caused by data transfer between memory and processing units.

To estimate the GPU memory required for serving an LLM [27], we refer to the formula

$$M = \frac{P \cdot 4B}{\left(\frac{32}{Q}\right)} \cdot 1.2$$

in which:

- M is the required GPU memory, expressed in Gigabyte
- P is the amount of (billion) parameters in the model
- 4B indicates 4 bytes, expressing the bytes used for each parameter
- 32, as there are 32 bits in 4 bytes

- Q (Precision or Size per parameter) is the amount of bits that should be used for loading the model (it can be 16, 8 or 4 bits)

- 1.2 represents a 20% overhead factor, needed for additional memory used during inference, such as storing activations (intermediate results) of the model

Thus, to determine the maximum size of the LLM supported by our GPU we can refer to the inverse formula

$$P = \frac{M \cdot 32}{1.2 \cdot Q \cdot 4B}$$

that, since I had an 8 Gigabyte GPU, leaded me to a 3.33B parameters model.

Fortunately, there is one approach to reduce GPU memory requirements for serving LLMs: quantization, a technique that decreases the precision of the model's parameters, by converting them from a higher to a lower precision format; this can significantly reduce memory usage, without a significant impact on accuracy.

In the previous formula, the corresponding parameter for the precision is Q that, for lower values, allows to run a 6.66B (Q=8) or 13.33B (Q=4) parameters model.

These values can vary, and, especially in tasks where the input includes not only text but also an image, a higher overhead coefficient must be taken into account.

As for the group of on-premise models, I chose to explore those available on Hugging Face Hub [28] (Fig. 3.11), a platform that allows users to share machine learning models and datasets by Hugging Face, Inc., an American company founded in 2016 by French entrepreneurs and based in New York City.

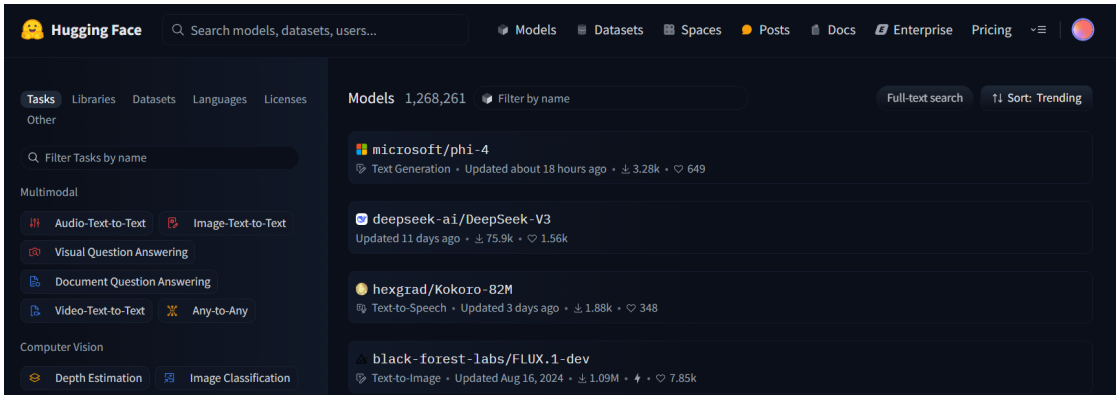

Figure 3.11: Hugging Face Hub interface

Initially aimed at developing a chatbot app for teenagers, the company now focuses on computational tools for building applications using machine learning.

The platform enables access to a vast number of models through its Transformers library, a Python package that provides open-source implementations of transformer models, compatible with multiple deep learning frameworks.

Among these models, in particular the ones referred to as image-text-to-text and also called vision-language models (VLMs), I selected:

1. LLaVA (Large Language and Vision Assistant) [29] [30], a large multimodal model (Fig. 3.12) combining the capabilities of:

   - a pre-trained CLIP (Contrastive Language–Image Pre-training) ViT-L/14 visual encoder to align images with textual descriptions, allowing the model to perform zero-shot learning across various visual tasks.

     This module processes input images (Xv) through Transformer layers to extract feature representations (Zv), enabling the model to interpret visual information efficiently; however, CLIP is mainly designed to establish high-level connections between images and text and does not inherently support deep reasoning or conversational interactions.

   - a Vicuna-1.5 (7B and 13B) language model (fϕ in the figure) on which LLaVA's linguistic capabilities rely, to achieve general-purpose visual and language understanding and generation capabilities.

     Vicuna processes and generates language responses (Xa) based on input language instructions (Xq), complementing the functionality of CLIP.

   - a Linear Projection, represented by a trainable matrix (W) acting as a bridge between the visual features (Zv) and the language model's embedding space: this transformation converts visual features into visual tokens (Hv), ensuring alignment with the language model's word embeddings and enabling seamless multimodal interaction.

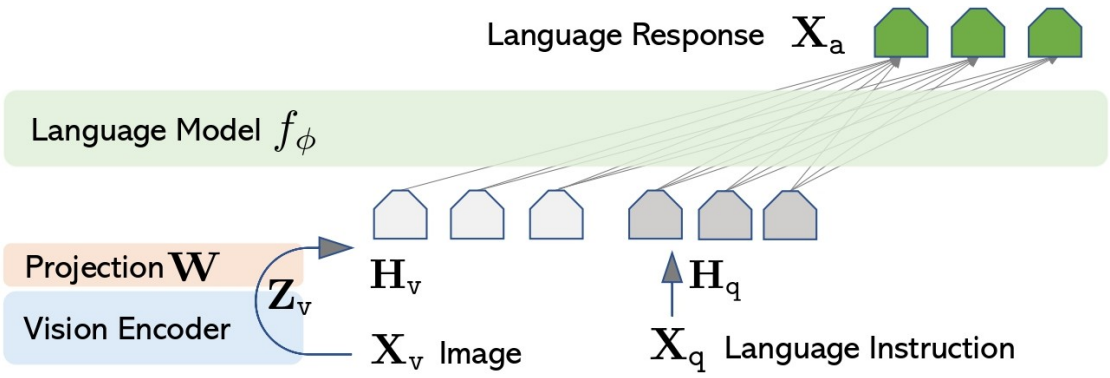

Figure 3.12: LLaVA (Large Language and Vision Assistant) architecture

Unlike conventional static datasets, LLaVA generates dynamic and instructive data using ChatGPT-4, actively incorporating this data into the training process to capture a broader range of human-like interactions: this enables LLaVA to engage in more complex reasoning and conversational tasks.

Following the release of the base LLaVA (1.5), LLaVA-NeXT (also known as LLaVA-1.6) [31] [32] was introduced as a model promising improved reasoning, OCR capabilities, and enhanced world knowledge.

LLaVA-NeXT incorporates a higher input image resolution using the "AnyRes" technique (Fig. 3.13), which segments the image into a grid of sub-images (or patches) with various configurations.

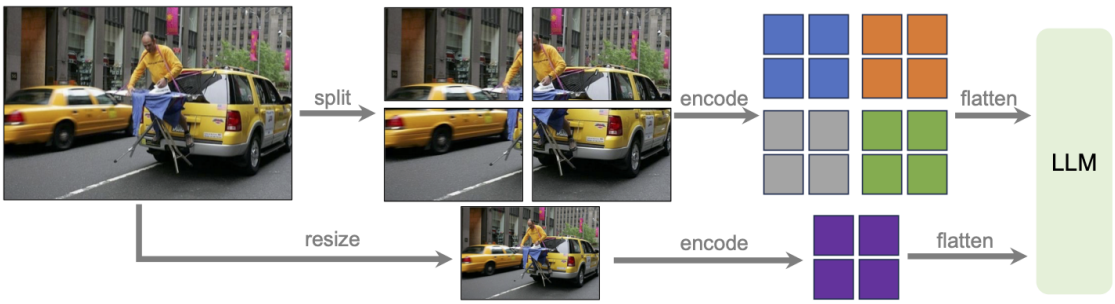

Figure 3.13: LLaVA-NeXT "AnyRes" technique

Additionally, alongside Vicuna, other LLMs were integrated including Mistral-7B and Nous-Hermes-2-Yi-34B, enabling LLaVA to address a broader range of users and scenarios effectively.

2. Phi-3.5-Vision [33], a 4.2B parameter model by Microsoft that excels in reasoning tasks and is adept at handling both single or multi-image and text prompts as inputs, subsequently generating textual outputs.

   It consists of two primary components:

   - a CLIP ViT-L/14 image encoder, the same used in LLaVA, which processes visual inputs
   - a Phi-3.5-mini Transformer decoder, which generates the model's textual outputs

   In Phi-3.5-Vision, the extracted visual tokens are combined with text tokens in an interleaved manner (Fig. 3.14), with no specific order required for image and text tokens while, to handle high-resolution images and various aspect ratios, a dynamic cropping strategy is employed, splitting the input image into a 2D array of blocks.

3. Idefics2 (Image-aware Decoder Enhanced à la Flamingo with Interleaved Cross-attentionS) [34], an open-access visual language model based on Flamingo, a language model developed by DeepMind, available in two different sizes (9B or 80B parameters).

   It accepts interleaved sequences of images and texts as input (Fig. 3.14) and generates coherent text as output, showing strong in-context few-shot learning capabilities.

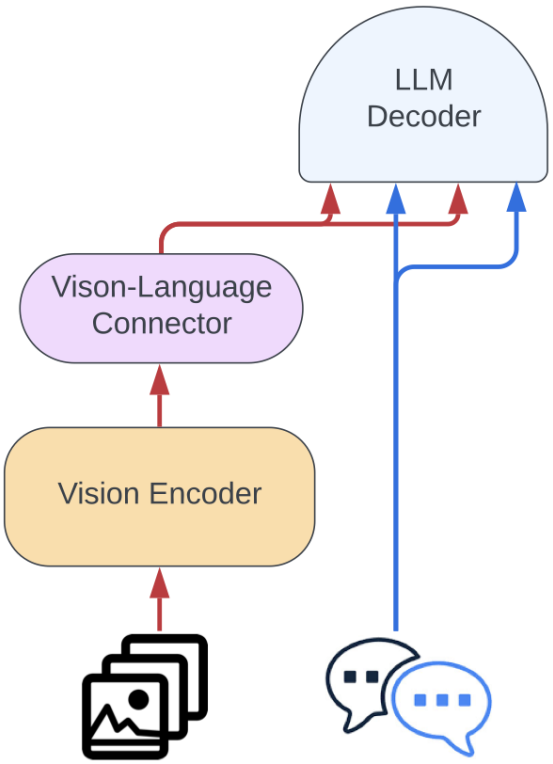

Figure 3.14: Generation of "interleaved" sequence of images and texts

Dataset

LLMs are trained on massive amounts of data, which makes them highly effective in zero-shot learning tasks, situations where at test time a model encounters samples from classes that were not observed during training, or more generally, when the distribution of training data differs from that of test data.

For this reason, training an LLM from scratch requires computational resources that very few possess; moreover, the wide availability of pre-trained open-source models makes this process unnecessary.

What is simpler to perform instead is fine-tuning, an approach to transfer learning where the parameters of a pre-trained neural network are adjusted on new data to specialize the model for a specific task of interest.

Fine-tuning can involve the entire neural network or only a subset of its layers: in the latter case, the layers that are not fine-tuned are said to be "frozen", meaning they are not updated during backpropagation.

Even so, fine-tuning requires datasets with at least a few thousand curated examples, which are often difficult to create unless they are already available.

What can be done, however, is experimenting with the model's textual input through prompt engineering, an iterative trial-and-error process used to structure the instructions given to the LLM (the prompt) to ensure it is better interpreted and generates the desired results.

Through this approach, requiring only a small test dataset consisting of a few hundred instances, I eliminated the need for a large dataset and built one consisting in 97 images, each with one or more value and unit of measurement couples (Fig. 3.15), for a total of 150 labels.

Figure 3.15: Example of an image for the digital gauge dataset

```
{
    "type": "digital",
    "values": [
        { "val": 20.0, "unit": "bar"},
        { "val": 100, "unit": "FS"}
    ]
}
```

Results

Since reading digital values is an "exact" method, it makes no sense to discuss about approximation errors or errors relatively to the measured value: for instance, in the example provided above, the measurement "700 FS" is considered just as incorrect as "108 FS".

To create a benchmark for the models, I decided to treat each "val-uom" pair as a string, focusing not on string similarity but on exact matches, obviously after a proper format standardization process.

In particular, I used the confusion matrix paradigm [35] (Fig. 3.16), commonly employed in classification tasks: a specific table layout visualizes the performance of a supervised algorithm by distinguishing real (actual) values from predicted ones, which can be either positive or negative.

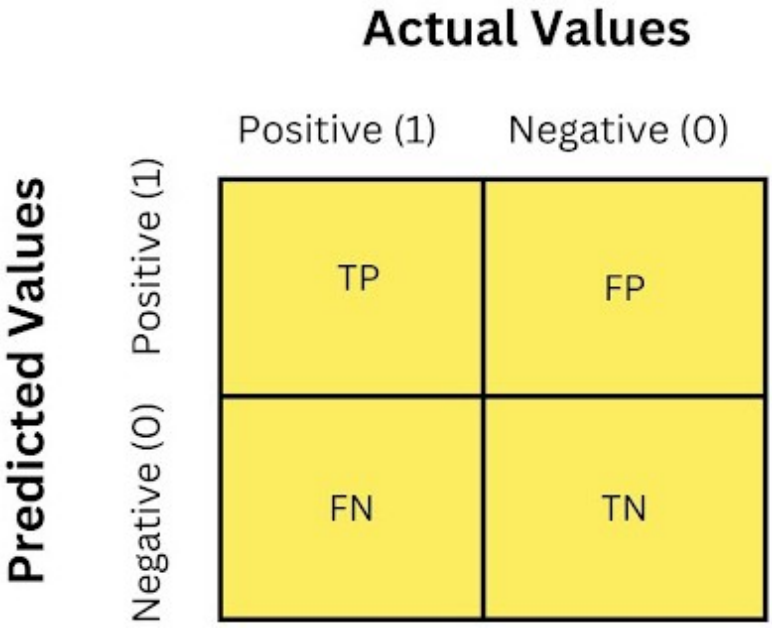

Figure 3.16: Confusion matrix in classification

Depending on these values, each instance is assigned to a corresponding portion of the matrix, allowing us to compute several performance metrics:

- Precision, the accuracy with which the model predicts positive classes, considering how many are actually positive, reliable when the cost of False Positives is high
- Recall (or Sensitivity), the ratio of correctly identified positive instances considering how many were predicted as positive, suitable when the cost of False Negatives is high

- F1 Score, the harmonic mean of Precision and Recall, particularly useful when seeking a balance between them, as it achieves a high score only when both metrics are high

In our case, positives and negatives will not indicate the presence or absence of specific classes but rather the presence of certain values.

Addictionally, since we aim to detect each value correctly without making multiple predictions for a single value, we will primarily rely on the F1 Score to obtain balanced results.

In Table 3.1, we can see the results of an initial evaluation of the models mentioned above.

Model	Precision (%)	Recall (%)	F1 Score (%)
GPT-4o	**76.46**	**79.89**	**77.26**
GPT-4o mini	70.06	73.30	71.08
GPT-4	64.95	66.22	64.93
LLaVA-1.5-Vicuna	2.10	5.67	2.99
LLaVA-1.6-Vicuna	3.99	10.31	5.52
LLaVA-1.6-Mistral	10.91	19.76	13.61
Phi-3.5-Vision	**16.44**	**24.97**	**18.70**
Idefics2	0.00	0.00	0.00

Table 3.1: LLMs comparison on digital gauge reading task

It is immediately evident that the large models from the first section achieve significantly better results compared to the smaller on-premise models from the second section.

They additionally boast a processing time of just 3 seconds, compared to an average of about 30 seconds for the on-premise models, due to computation on local hardware.

This outcome was expected, as the various GPT models were considered a strong baseline to numerically quantify a gap that we already knew existed.

Despite this, the next step was to identify and test approaches that could bridge this gap, including:

- providing the image as input to a traditional OCR model, then extending the LLM's prompt with the OCR's output: to this end I choose EasyOCR [36], a Python package for detecting and extracting text from images, particularly flexible and easy to use

- converting the image to grayscale before passing it to the LLM as it is well known that OCR models, whose capabilities are often embedded in LLMs, benefit from variations in the chromatic scale [37] (Fig. 3.17)

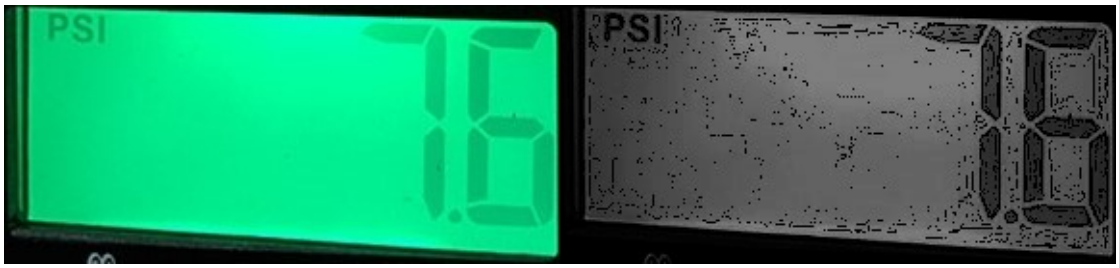

Figure 3.17: Image preprocessing for OCR improvement

At this point, I decided to focus only on the two best-performing models from Table 3.1, as the others either failed to provide acceptable results in terms of scores or were unable to produce outputs in the required format.

In Table 3.2, we can observe the results of the potentially improving approaches discussed earlier, from which it is evident how performing an image pre-processing helps the model read the display.

Model	Precision (%)	Recall (%)	F1 Score (%)
Phi-3.5-Vision	16.44	24.97	18.70
Phi-3.5-Vision (OCR)	11.26	14.94	12.08
Phi-3.5-Vision (grayscale)	**23.07**	**33.01**	**25.60**
LLaVA-1.6-Mistral	10.91	19.76	13.61
LLaVA-1.6-Mistral (OCR)	7.15	11.34	8.41
LLaVA-1.6-Mistral (grayscale)	**15.31**	**26.12**	**18.63**

Table 3.2: Possible improvements on digital gauge reading task

In conclusion, this section aimed to analyze the performance of various multimodal models, both on-premise and cloud-based, quantifying the numerical gap between the two categories; this analysis enabled the proposal of various alternatives to clients, each with its own pros and cons.

Possible future improvements could include the use of local models with a larger number of parameters, enabled by upgrades to the available hardware, or the exploration of other pre-processing techniques and more advanced image-to-text models like GOT-OCR2.0 [38].

3.1.2 Analog gauges

Even though most of the useful data available today has already been digitized, a surprising number of gauges and meters remain analog [39].

In fact, analog gauges are designed to be easily readable by humans, but they still pose significant challenges for even state-of-the-art computer vision algorithms because of the wide variety of gauge designs, units, and measurement scales.

The most obvious solution to this issue is to replace analog dials with digital sensors and displays; however, this is not always feasible due to cost, compatibility issues, or constraints imposed by equipment or manufacturers.

Previous methodology

To read analog gauge measurements, the company had already developed a pipeline based on [40], consisting of two phases:

1. Calibration, in which were involved computer vision models specifically trained to recognize the center of the dial and the tip of the needle (Fig. 3.18).

Figure 3.18: Calibration phase in previous methodology

2. Measurement, where they used the obtained information to calculate the angle of the needle and identify the correlation between the dial's angle and the value it pointed to (Fig. 3.19).

Figure 3.19: Measurement phase in previous methodology

However this application, involving a considerable amount of trigonometry, had the limitation of requiring the user to manually provide the unit of measurement, as well as the range of values and angles on the scale in the format

```
{
    "min_angle": 45,
    "max_angle": 315,
    "min_value": 0,
    "max_value": 200,
    "unit": "psi"
}
```

Without these informations, it was impossible to perform any analysis: for this reason, I tried finding an alternative solution that would be more generalizable.

New proposed approach

Based on [41], I adopted a method that requires no prior knowledge of the gauge type, the range of the scale or the units used, as it is capable of extracting such information automatically.

To propose a robust and modular algorithm, this new approach relies on a set of reasonable assumptions:

- the gauge has a linear scale with at least 5 visible major notches aligned in an arc on the gauge face
- there are numerical scale markers along the scale, and the gauge face is flat
- the midpoint between the start and end notches of an unrotated gauge is positioned at the bottom
- the needle points towards the symmetry axis of the needle

This method splits the reading task into distinct steps, also enabling the detection of potential failures at each stage.

1. Gauge detection

   As a first step, the method detects (Fig. 3.20a) the gauge, crops (Fig. 3.20b) and resize it to a 448x448 resolution, to isolate individual gauges, if multiple are present, while also reducing background noise.

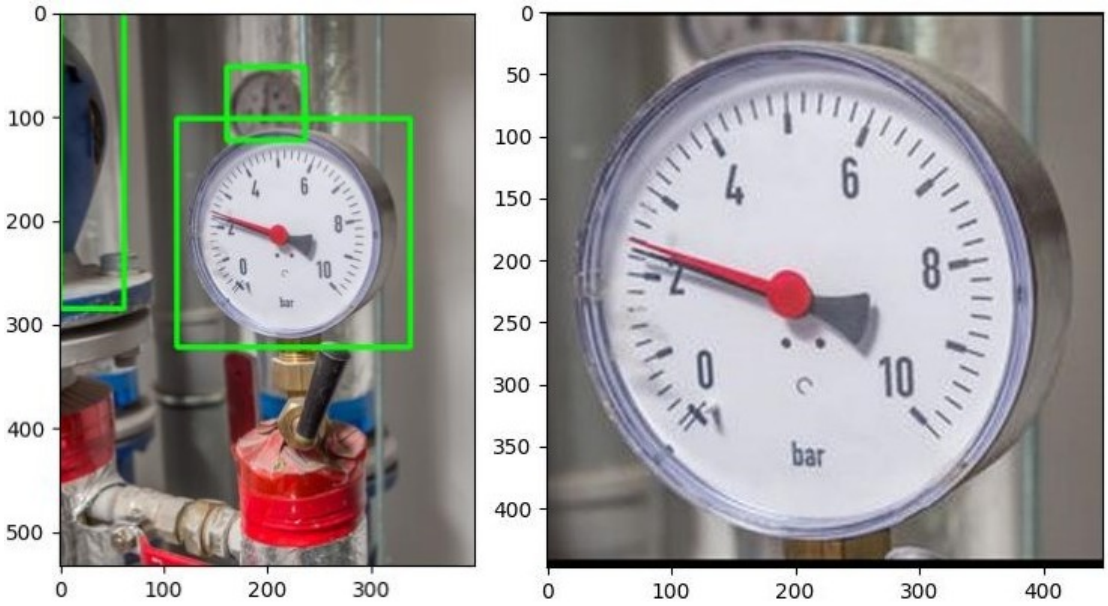

Figure 3.20: Gauge detection and crop

   To achieve this, the smallest YOLOv8 detection model was used, as it proved more than sufficient for this task.

   Initially pretrained on the COCO dataset, the model was further fine-tuned on a small, task-specific dataset.

2. Notch detection

   The method detects the major notches on the gauge scale using a keypoint detection algorithm, that leverages a pretrained DINOv2 vision transformer backbone to predict a heatmap (Fig. 3.21), where each pixel's intensity corresponds to the probability of it being a notch.

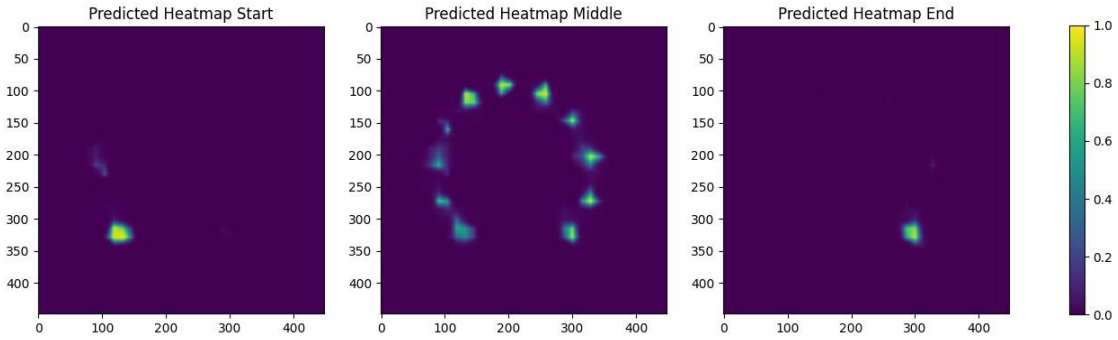

Figure 3.21: Heatmap prediction and clustering

   The heatmap is then clustered to extract individual notch points, which are categorized into three types (Fig. 3.22): Start notch, Intermediate notches, and End notch.

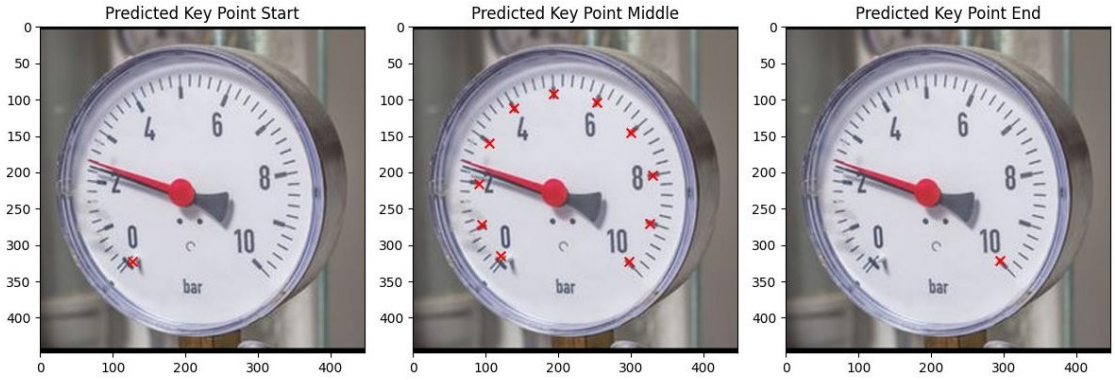

Figure 3.22: Notch detection and categorization

   The start and end notches are particularly important, as they are used to estimate and, if necessary, correct the orientation of the gauge in a later stage of the pipeline.

3. Ellipse fitting

   To approximate the arc of the gauge scale, an ellipse is fitted through the coordinates of the keypoints corresponding to the detected notches (Fig. 3.23), with a minimum of 5 required.

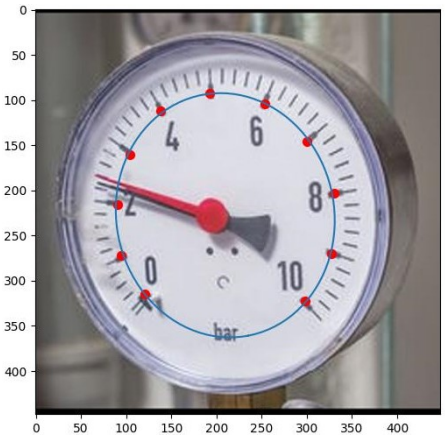

Figure 3.23: Ellipse fitting through keypoints

This step is crucial for correcting perspective distortions by warping the gauge image so that the computed ellipse is transformed into a perfect circle.

4. Needle segmentation and line fit

   The task of estimating the position and direction of the gauge needle is addressed in two steps (Fig. 3.24):

   - pixels belonging to the needle are segmented using the same COCO-pretrained (and fine-tuned) YOLOv8 model size
   - a line is fitted through the segmented pixels applying Orthogonal Distance Regression, to minimize the shortest distance of each point to the line

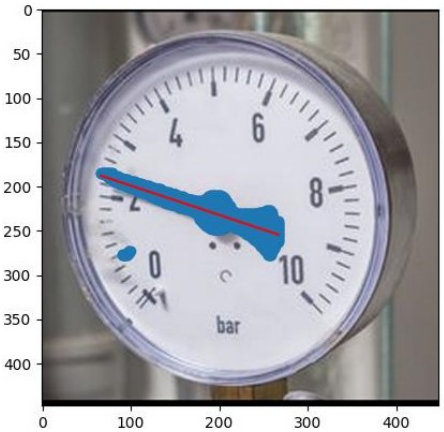

Figure 3.24: Needle pixels segmentation and line fitting

5. Detection of scale numbers and unit

   Since the algorithm does not have prior knowledge of the scale's range, it extracts the numbers (at least two scale markers are required) and the unit displayed on the gauge face using an OCR model (Fig. 3.25).

   The OCR process relies on a two-stage pre-trained neural network in which a DBNet model [42] with a ResNet-18 backbone focuses on text detection, while an ABINet architecture [43], which combines a vision model and a language model, predicts the text contained within the detected bounding boxes.

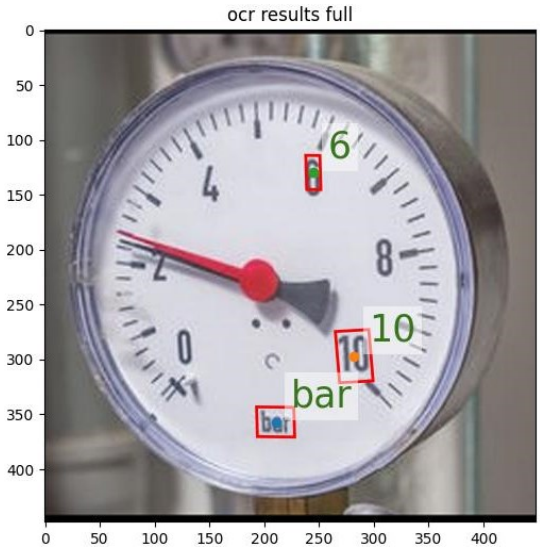

Figure 3.25: Detection and extraction of scale numbers and unit

6. Computing the final reading

   After running the previous stages of the pipeline, it computes the reading of the gauge using the following steps:

   - **Projection and interpolation of scale markers**

     From the text recognized during the OCR stage, the algorithm selects only the subset of recognitions that can be interpreted as numerical values, projecting the center of the corresponding bounding boxes onto the ellipse to anchor the markers and determine the needle angle for each marker.

     Using these angles, the algorithm fits a correspondence line that maps the angles on the ellipse to the scale readings (Fig. 3.26) using linear regression with least squares.

     Moreover to handle outliers, such as faulty OCR readings, it employs RANSAC (Random Sample Consensus) to identify and remove them.

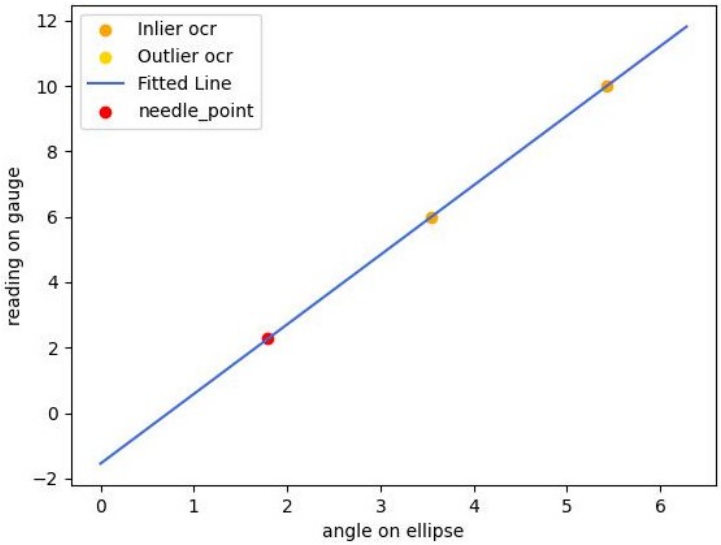

Figure 3.26: Projection and interpolation of scale markers

For gauges with multiple scales, the model distinguishes between markers for inner and outer scales by checking whether the it lies inside or outside the ellipse: once this distinction is made, the algorithm fits separate linear models for each scale and calculates the corresponding reading.

- **Intersection of needle and ellipse**
  The final step involves determining the point on the ellipse that the needle is pointing to, by finding the intersection of the needle line and the estimated ellipse; then the algorithm computes the angle of the point on the ellipse and predicts the reading (Fig. 3.27) by performing interpolation with the previously fitted linear model that maps angles to readings.

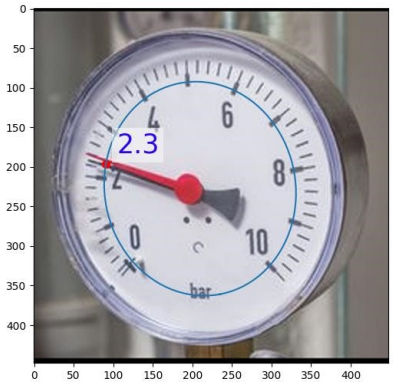

Figure 3.27: Intersection of needle line and estimated ellipse

Dataset

Since the models utilized in the new methodology were already pre-trained, there was no need for a large dataset: a small test dataset was more than sufficient to evaluate and validate the new approach.

To this end, I created a manually labeled dataset consisting of 63 images of analog gauges, including both single-scale and dual-scale gauges (Fig. 3.28), following a standard format.

Figure 3.28: Example of an image for the analog gauge dataset

```
{
    "type": "analog",
    "values": [
        { "val": 6, "range": 25, "unit": "bar"},
        { "val": 90, "range": 360, "unit": "psi"}
    ]
}
```

Results

In this case, unlike digital gauges, where an "exact" method was used, it makes sense to talk about approximation errors and errors relative to the measurement scale's range.

This is because the final result is not derived from an attempt to read numerical characters by a Generative AI model, but is instead the output of a complex

Computer Vision pipeline that accumulates a certain degree of approximation error at each step.

To establish a benchmark, I decided to define various percentage tolerance levels relative to the total range of the gauge scale: 5, 10, 20 and "infinite" (∞), that represents the algorithm's ability to successfully reach the final stage of the pipeline and return a value, regardless of its accuracy.

Alongside the previous methodology, which without prior knowledge of the gauge is incapable of returning any value, I included the new algorithm in its original form as well as a couple of its variants testing different architectures, specifically for the optical character recognition (OCR) step, which has proven to be the most critical stage of the pipeline.

Specifically, for both variants, instead of using the DBNet model with a ResNet-18 backbone for text detection, I opted for a DBNetpp [44] with a ResNet-50 backbone, to enhance the model's capabilities.

Regarding text recognition, I continued to use ABINet for variant 1, while for variant 2 I replaced it with ABINet-Vision, which isolates the vision model component of ABINet.

Methodology	∞ tol. (%)	20 tol. (%)	10 tol. (%)	5 tol. (%)
Old	0.0	0.0	0.0	0.0
New (original)	53.3	45.0	41.7	35.0
New (variant 1)	**71.7**	**61.7**	**55.0**	48.3
New (variant 2)	**71.7**	60.0	**55.0**	**50.0**

Table 3.3: Methodology comparison on analog gauge reading task

From Table 3.3 it is evident that, compared to the original model, both variants achieve approximately a 15% improvement at each tolerance level, allowing for the reading of more gauges while also reducing relative error.

In conclusion, this section introduced a reading method that aims to achieve robust performance across gauges found in real-world scenarios with diverse appearances, scales, and units, as it does not require any prior information.

Possible future improvements could include fine-tuning the OCR reader on text specific to gauges, identifying other useful pre-processing steps, or even combining readings from different images.

3.2 Autonomous surveillance on restricted areas

Context and motivation

Ensuring the security and operational integrity of industrial facilities, critical infrastructures, and restricted areas is a major challenge across various sectors, including manufacturing, energy, logistics, and defense.

Traditional surveillance methods rely heavily on fixed cameras, security personnel, and scheduled inspections, which often come with limitations in coverage, response time, and human resource efficiency.

Modern industrial sites require a more dynamic, intelligent, and autonomous approach to surveillance, capable of detecting anomalies, unauthorized access, and potential hazards in real time.

Specifically, for this second task, the objective was to detect the presence of unauthorized individuals and vehicles within a restricted-access area (Fig. 3.29), which could belong to an industrial facility or potentially even to a private property.

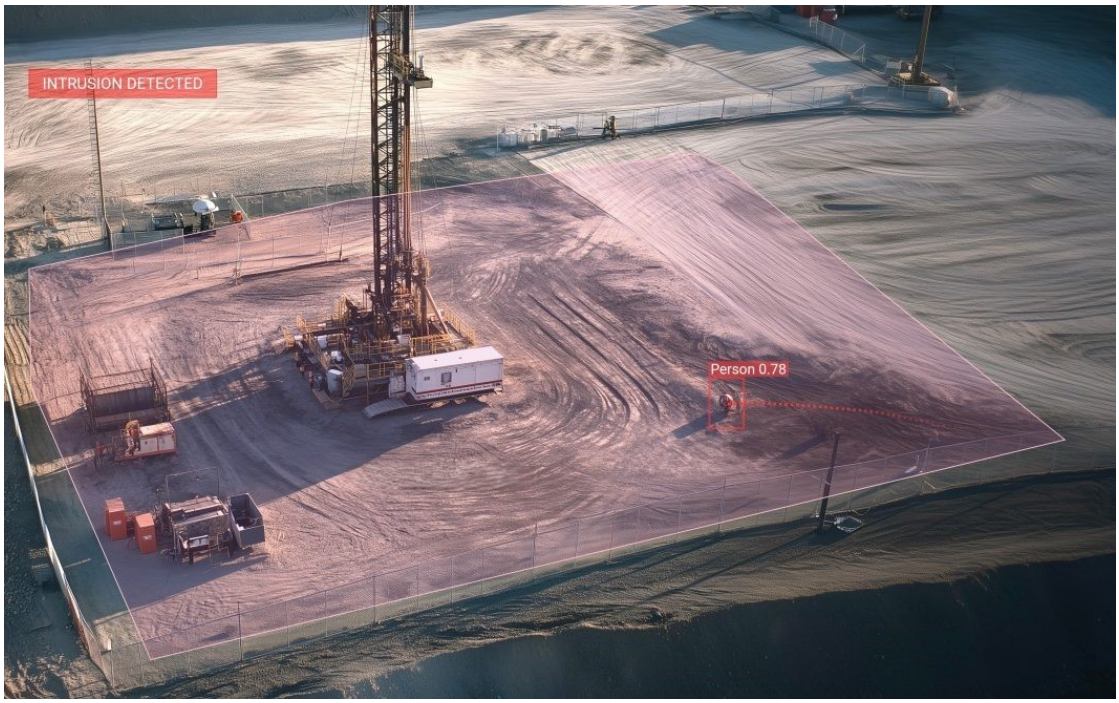

Figure 3.29: Example of surveillance on a region of interest (ROI)

Compared to traditional approaches, these type of systems aim to offer greater flexibility, wider coverage, and the ability to operate in hazardous or hard-to-reach areas without human intervention.

Setup

For this use case, a video source with broad-spectrum visibility was required, preferably from an elevated position, so a standard surveillance camera could have been suitable.

But given the large area to be monitored, the alternatives were either installing multiple cameras or using a source capable of covering vast distances quickly: for this reason, the choice fell on the previously described DJI drone.

Additionally, to enable the autonomous execution of multiple missions, we used the DJI Dock 2 [45] (Fig. 3.30), a hangar compatible with many DJI drones, that is recognized by the drone through positioning markers on the landing platform, allowing for precise landings without the need for direct human control.

Figure 3.30: DJI's hangar, Dock 2

The Dock 2 is designed to operate reliably for extended periods, even in harsh climates and environments, and integrates various sensors, including rain, wind speed, and temperature indicators, to detect real-time weather changes: these, along with online weather forecasts, can provide timely alerts or stop flight operations if conditions become hazardous.

The final implementation includes the integration of a system that monitors a dedicated email inbox in real time, specifically created to receive alerts from the alarm system in case of intrusions; based on the email's content (Fig. 3.31), the system can extract information about the specific breached area and immediately launch the corresponding autonomous drone mission to that area.

Once deployed for surveillance, the drone will search for individuals, and if any are detected, it will capture images and send them (along with other relevant

Figure 3.31: Template of email triggering drone missions

information such as location) to a list of authorized recipients, including the property owner, the security company, or law enforcement authorities.

Only when the inspection is complete, the drone can return to its initial position, ready for the next mission.

Dataset

Treating it as a detection task, I searched for publicly available pre-labeled datasets, as manually labeling such data would be extremely costly and time-consuming.

Among the various options available on platforms like Roboflow, I chose to use VisDrone [46] (Fig. 3.32), a large-scale benchmark captured by various drone cameras available for different computer vision tasks, covering a wide range of aspects, including location, environment (both urban and rural), objects (pedestrians, cars, bicycles, etc.), and density (ranging from sparse to crowded scenes).

Figure 3.32: A VisDrone dataset labelled instance

Results

By comparing the areas of the ground truth and predicted bounding boxes, it was possible to use the previously mentioned metrics [47], along with:

- mAP50: Mean Average Precision calculated at an Intersection over Union (IoU) threshold of 0.50, representing accuracy based only on the "easy" detections
- mAP50-95: The average Mean Average Precision calculated across varying IoU thresholds, ranging from 0.50 to 0.95

Table 3.4 presents the results of an initial training performed using a YOLO model of a reference version and size, while keeping the class definitions from the VisDrone dataset unchanged.

Setup	Precision	Recall	mAP50	mAP50-95
All_classes	0.47955	0.34217	0.35128	0.20542

Table 3.4: Results from the first train on VisDrone detection dataset

By closely analyzing the confusion matrix and the distribution of instances within the dataset (Fig. 3.33), it became evident that the model struggles with the presence of multiple sub-classes of the main categories of people and vehicles.

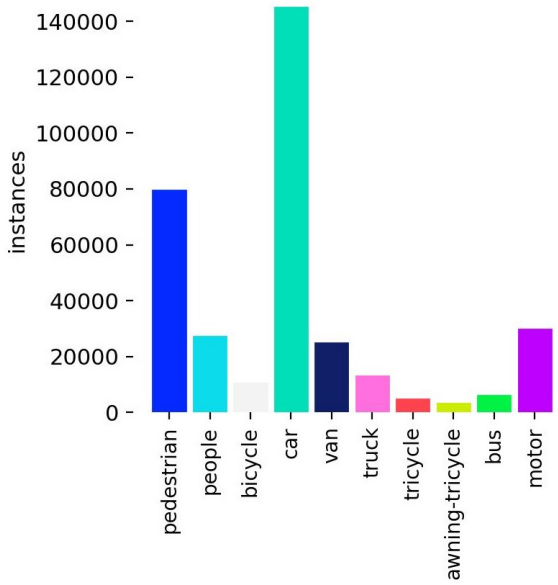

Figure 3.33: Distribution of instances across classes in VisDrone dataset

For this reason, I decided to merge these sub-classes into broader categories, specifically creating the macro-classes "person" and "vehicle", for which Table 3.5 presents the results of a second training using this new categorization.

Setup	Precision	Recall	mAP50	mAP50-95
All_classes	0.47955	0.34217	0.35128	0.20542
All_classes_merged	0.69427	0.49799	0.57311	0.31330

Table 3.5: Results from the second train on VisDrone detection dataset

While it is evident an improvement in results, the metrics curves (Fig. 3.34) reveal that the model tends to specialize heavily in detecting vehicles (in red), while neglecting people (in blue).

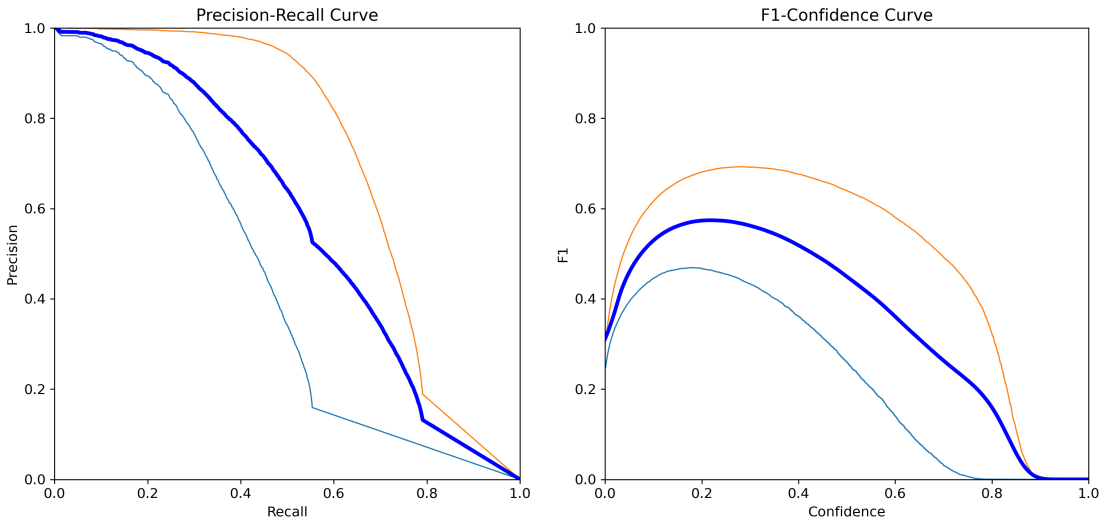

Figure 3.34: Validation curves in VisDrone dataset

At this point, I decided to train two separate models for the two macro-classes to determine whether the lower performance on people was due to the inherent difficulty of the class, since they are harder to recognize due to their smaller size, or if it was primarily a consequence of the imbalance in the number of instances compared to vehicles.

Setup	Precision	Recall	mAP50	mAP50-95
All_classes	0.47955	0.34217	0.35128	0.20542
All_classes_merged	0.69427	0.49799	0.57311	0.31330
Only_persons_merged	0.66056	0.47518	0.53231	0.22796
Only_vehicles_merged	0.8217	0.60716	0.71128	0.4445

Table 3.6: Results from the third train on VisDrone detection dataset

From the results in Table 3.6, we can conclude that, to maximize evaluation metrics for a single model, it would be beneficial to expand the dataset to balance the two macro-classes and, additionally, comparing different versions and sizes of the YOLO model while carefully considering the trade-off between accuracy and performance.

Chapter 4

Conclusion

4.1 Challenges and Opportunities

This thesis explored the integration of Robotics, Computer Vision, and GenAI in industrial automation through an interdisciplinary approach that leverages state-of-the-art technologies, combined with advanced AI models, to improve efficiency, accuracy, and reliability in industrial monitoring.

1. In the domain of **automated gauge reading**, we offered a scalable solution for real-time monitoring of both digital and analog gauge data, while also revealing challenges such as dataset limitations and model performance variations based on gauge design.

   Future improvements could focus on expanding training datasets, refining OCR models, and integrating additional pre-processing techniques to enhance accuracy.

2. For **autonomous surveillance of restricted areas**, we designed and tested a vision-based anomaly detection system using UAVs and Computer Vision models, offering an alternative to traditional surveillance methods.

   Future research could involve improving model generalization, and advancements in low-power AI hardware for onboard inference, as real-time processing constraints remain areas for further development.

Overall, this thesis has demonstrated that the convergence of Robotics, Computer Vision, and GenAI holds immense promise for revolutionizing industrial automation: while challenges persist, the continued evolution of AI models and robotics platforms will likely unlock new capabilities, paving the way for more intelligent, autonomous, and efficient industrial monitoring systems.

Bibliography

[1] Sprint Reply. *Soluzioni AI per l'efficienza aziendale*. Website. URL: https://www.reply.com/sprint-reply/it (cit. on p. 2).

[2] Area42. *Il centro di sviluppo Reply dove l'innovazione tecnologica diventa reale*. Website. URL: https://www.reply.com/it/labs/area-42 (cit. on p. 2).

[3] Boston Dynamics. *Changing your idea of what robots can do.* Website. URL: https://bostondynamics.com/ (cit. on p. 5).

[4] Wikipedia contributors. *Boston Dynamics*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Boston_Dynamics (cit. on p. 5).

[5] Boston Dynamics. *Spot - The Agile Mobile Robot*. Website. URL: https://bostondynamics.com/products/spot/ (cit. on p. 6).

[6] Wikipedia contributors. *DJI*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/DJI (cit. on p. 8).

[7] DJI Store Italia. *DJI Matrice 3TD - Industrial Drone*. Website. URL: https://www.dji-store.it/prodotto/dji-matrice-3td/ (cit. on p. 8).

[8] Ultralytics. *Everything You Need to Know About Computer Vision in 2025*. Blog post. URL: https://www.ultralytics.com/blog/everything-you-need-to-know-about-computer-vision-in-2025 (cit. on p. 10).

[9] Ultralytics. *Ultralytics Tasks Documentation*. Online Documentation. URL: https://docs.ultralytics.com/tasks/ (cit. on p. 10).

[10] Ultralytics. *Ultralytics - AI Vision Solutions*. Website. URL: https://www.ultralytics.com/ (cit. on p. 13).

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: https://arxiv.org/abs/1506.02640 (cit. on p. 13).

[12] Nikhil Rao. *YOLOv11 Explained: Next-Level Object Detection with Enhanced Speed and Accuracy*. Medium. URL: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71 (cit. on p. 15).

[13] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV]. URL: https://arxiv.org/abs/2304.02643 (cit. on p. 17).

[14] Nikhila Ravi et al. *SAM 2: Segment Anything in Images and Videos*. 2024. arXiv: 2408.00714 [cs.CV]. URL: https://arxiv.org/abs/2408.00714 (cit. on p. 18).

[15] Wikipedia contributors. *Generative artificial intelligence*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Generative_artificial_intelligence (cit. on p. 20).

[16] Wikipedia contributors. *Long short-term memory*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Long_short-term_memory (cit. on p. 22).

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on p. 24).

[18] Wikipedia contributors. *Seq2seq*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Seq2seq (cit. on p. 24).

[19] Inside Machine Learning. *What is a Transformer?* Medium. URL: https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04 (cit. on p. 25).

[20] Abdulkader Helwan. *ELMo: Deep Contextualized Word Representations*. Medium. URL: https://abdulkaderhelwan.medium.com/elmo-e-c8e6fc068882 (cit. on p. 28).

[21] Wikipedia contributors. *BERT (language model)*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/BERT_(language_model) (cit. on p. 28).

[22] Tenyks Blogger. *Multimodal Large Language Models (MLLMs): Transforming Computer Vision*. Medium. URL: https://medium.com/@tenyks_blogger/multimodal-large-language-models-mllms-transforming-computer-vision-76d3c5dd267f (cit. on p. 31).

[23] Wikipedia contributors. *OpenAI*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/OpenAI (cit. on p. 32).

[24] Wikipedia contributors. *Generative pre-trained transformer*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Generative_pre-trained_transformer (cit. on p. 32).

[25] Midwest Instrument. *Why Are Pressure Gauges Important?* Midwest Instrument Blog. 2022. URL: https://midwestinstrument.com/2022/04/28/why-are-pressure-gauges-important/ (cit. on p. 35).

[26] Wikipedia contributors. *Optical character recognition*. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Optical_character_recognition (cit. on p. 40).

[27] K. Singh. *Calculate How Much GPU Memory You Need to Serve Any LLM*. Medium. URL: https://ksingh7.medium.com/calculate-how-much-gpu-memory-you-need-to-serve-any-llm-67301a844f21 (cit. on p. 41).

[28] Hugging Face. *Hugging Face Models Hub*. Website. URL: https://huggingface.co/models (cit. on p. 42).

[29] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. *Visual Instruction Tuning*. 2023. arXiv: 2304.08485 [cs.CV]. URL: https://arxiv.org/abs/2304.08485 (cit. on p. 43).

[30] Uddeshya. *Introduction to LLaVA: A Multimodal AI Model*. Medium. URL: https://medium.com/@ud.uddeshya16/introduction-to-llava-a-multimodal-ai-model-2a2fa530ace4 (cit. on p. 43).

[31] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. *Improved Baselines with Visual Instruction Tuning*. 2023 (cit. on p. 44).

[32] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. *LLaVA-NeXT: Improved reasoning, OCR, and world knowledge*. Jan. 2024. URL: https://llava-vl.github.io/blog/2024-01-30-llava-next/ (cit. on p. 44).

[33] Marah Abdin et al. *Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone*. 2024. arXiv: 2404.14219 [cs.CL]. URL: https://arxiv.org/abs/2404.14219 (cit. on p. 44).

[34] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. *What matters when building vision-language models?* 2024. arXiv: 2405.02246 [cs.CV]. URL: https://arxiv.org/abs/2405.02246 (cit. on p. 45).

[35] Nirajan Acharya. *Understanding Precision, Recall, F1-Score, and Support in ML Evaluation*. Medium. URL: https://medium.com/@nirajan.acharya777/understanding-precision-recall-f1-score-and-support-in-machine-learning-evaluation-7ec935e8512e (cit. on p. 47).

[36] Roboflow. *How to Use EasyOCR*. Roboflow Blog. URL: https://blog.roboflow.com/how-to-use-easyocr/ (cit. on p. 48).

[37] Marius M. Şerban, Stefan Ruseti, Radu Tudor Ionescu, Mădălina Uricaru, and Mihai Petrovici. «OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader and Transym». In: *ResearchGate* (2016). URL: https://www.researchgate.net/publication/310645810_OCR_as_a_Service_An_Experimental_Evaluation_of_Google_Docs_OCR_Tesseract_ABBYY_FineReader_and_Transym (cit. on p. 49).

[38] Haoran Wei et al. *General OCR Theory: Towards OCR-2.0 via a Unified End-to-end Model*. 2024. arXiv: 2409.01704 [cs.CV]. URL: https://arxiv.org/abs/2409.01704 (cit. on p. 49).

[39] Roboflow. *Read Analog Dials with Computer Vision*. Roboflow Blog. URL: https://blog.roboflow.com/read-analog-dials-computer-vision/ (cit. on p. 50).

[40] Intel IoT DevKit. *Analog Gauge Reader*. GitHub Repository. URL: https://github.com/intel-iot-devkit/python-cv-samples/blob/master/examples/analog-gauge-reader/analog_gauge_reader.py (cit. on p. 50).

[41] Maurits Reitsma, Julian Keller, Kenneth Blomqvist, and Roland Siegwart. *Under pressure: learning-based analog gauge reading in the wild*. 2024. arXiv: 2404.08785 [cs.CV]. URL: https://arxiv.org/abs/2404.08785 (cit. on p. 51).

[42] Minghui Liao, Zhaoyi Wan, Cong Yao, Kai Chen, and Xiang Bai. *Real-time Scene Text Detection with Differentiable Binarization*. 2019. arXiv: 1911.08947 [cs.CV]. URL: https://arxiv.org/abs/1911.08947 (cit. on p. 55).

[43] Shancheng Fang, Hongtao Xie, Yuxin Wang, Zhendong Mao, and Yongdong Zhang. *Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition*. 2021. arXiv: 2103.06495 [cs.CV]. URL: https://arxiv.org/abs/2103.06495 (cit. on p. 55).

[44] Minghui Liao, Zhisheng Zou, Zhaoyi Wan, Cong Yao, and Xiang Bai. *Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion*. 2022. arXiv: 2202.10304 [cs.CV]. URL: https://arxiv.org/abs/2202.10304 (cit. on p. 58).

[45] DJI Enterprise. *DJI Dock 2*. Website. URL: https://enterprise.dji.com/it/dock-2 (cit. on p. 60).

[46] Pengfei Zhu, Longyin Wen, Xiao Bian, Haibin Ling, and Qinghua Hu. *Vision Meets Drones: A Challenge*. 2018. arXiv: 1804.07437 [cs.CV]. URL: https://arxiv.org/abs/1804.07437 (cit. on p. 61).

[47] Ultralytics. *YOLO Performance Metrics Guide*. Ultralytics Documentation. URL: https://docs.ultralytics.com/guides/yolo-performance-metrics/ (cit. on p. 62).