# Deep Learning Based Object Recognition Approach for Motorbike Dashboard Software Validation

submitted as a project report

**Master's degree program**
**Automotive Mechatronics and Management**

to obtain the academic degree of
**Master of Science in Engineering (MSc)**

by
**Seifeldien Hussein**

Project supervisor:
**FH-Prof. DI Mag. Dr. techn. Thomas Schlechter MA**

**January 2025**

# Contents

# Acronyms

| | |
|---|---|
| **AFPM** | Adaptive Feature Pyramid Module. 22 |
| **AI** | Artificial Intelligence. 8, 11, 12, 20, 30 |
| **CL** | Convolutional Layer. 13–15 |
| **CNN** | Convolutional Neural Network. 13–18, 21, 25–27, 29, 30 |
| **CS** | Camera System. 2–4, 7, 23, 27, 28 |
| **DB** | Dash Board. 2, 3, 6, 7, 26, 30 |
| **DL** | Deep Learning. 2, 3, 7, 8, 11, 12, 25–28, 30 |
| **DNN** | Deep Neural Network. 11, 12 |
| **DPM** | Deformable Part-Based Model. 10, 11 |
| **DR** | Dynamic Range. 23, 27, 28 |
| **FCL** | Fully Connected Layer. 13, 15, 17–19 |
| **FOV** | Field Of View. 23, 24 |
| **FPS** | Frames Per Second. 19, 20, 22, 30 |
| **GPU** | Graphical Processing Unit. 13 |
| **HDR** | High Dynamic Range. 26–28 |
| **HIL** | Hardware In The Loop. 2, 3 |
| **HOG** | Histograms Of Oriented Gradients. 9–11 |
| **LCFM** | Light-weight Classification Function Module. 22 |

**MAP** Mean Average Precision. 19, 22

**ML** Machine Learning. 10–12

**NMS** Non Maximum Suppression. 10, 16, 19, 20

**NN** Neural Network. 12, 13

**OD** Object Detection. 2–4, 7–11, 15, 16, 18–20, 22, 24, 25, 27–30

**OEM** Original Equipment Manufacturer. 2, 3

**PL** Pooling Layer. 13–15

**RCNN** Region Based Convolutional Neural Network. 16–18, 27

**ROI** Region Of Interest. 8, 16, 17

**SDR** Standard Dynamic Range. 27, 28

**SPP-Net** Spatial Pyramid Pooling Networks. 16–18

**SSD** Single Shot MultiBox Detector. 19, 20, 27

**SVM** Support Vector Machine. 10, 12, 16

**SW** Software. 2–6, 23

**VJ** Viola Jones. 9

**YOLO** You Only Look Once. 18–20

# 1. Technical Background

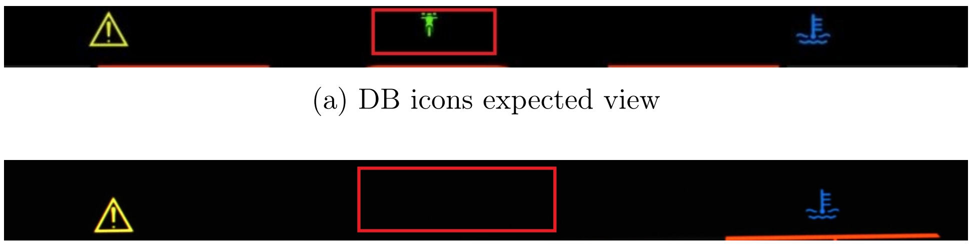
This chapter explores the essential elements of testing and Object Detection (OD), highlighting their role in Software (SW) development and validation. It begins with an overview of motorbike Dash Board (DB) and the need for reliable testing methods to ensure safety and functionality. The chapter then introduces OD techniques and how they are used in DB testing in a Hardware In The Loop (HIL) setup. The discussion then shifts to testing, including its importance, methodologies, and levels, and provides a comparison between manual and automated testing techniques. Additionally, the chapter discusses OD, focusing on both traditional and Deep Learning (DL) based approaches, and their application in testing. Finally, it addresses key considerations for selecting Camera System (CS)s and their influence on the performance of OD algorithms.

## 1.1 Dashboard Testing

With the current shift in the motorbike industry toward the direction of SW defined vehicle, DB SW that provides the rider with a graphical user interface to monitor and control the bike has gained a huge attention from Original Equipment Manufacturer (OEM) in the past years to continuously provide a seamless and safe ride experience. As a result, the DB market is expected to grow from 10.71$ billion to 18.64$ billion in the time frame from 2021 to 2028 with a compound annual growth rate of 8.2% [1]. Current DBs not only provide the rider with hardware information such as speed, fuel level and diagnostics but also allow users to interact with the system in real-time through multiple advanced methods like touch, voice control and even hand gesture control. Riders depend on these systems for essential functions, including smart navigation, connectivity, and even advanced rider assistance.

Given these factors, the influence of DB SW on bike control, OEMs revenue streams, and rider safety continues to grow year after year. Furthermore, any bugs in the DB SW could cause dangerous consequences, for instance, vehicle recalls, a decline in consumer confidence, and even potential loss of lives. That is why a very effective and efficient SW testing technique should be implemented in order to detect and eliminate any SW defects and be able to ensure that the SW and features were developed according to the requirements defined by the OEMs.

Design defects can arise from systematic issues, such as pixel defects, uneven screen uniformity, or signal conversion errors. Additionally, they may result from human errors by designers or developers, leading to typographical mistakes, rendering issues in the text, or the display of incorrect icons, as shown in Figure 1.1, the expected view from the photoshop file (Figure 1.1a) and the actual DB view captured using a camera (Figure 1.1b) shows an example of incorrect icons or loss of asset due to a DB SW error. One can see that there is a difference in the color scheme between the captured image and photoshop image but this will be treated in the later sections. Currently, the automotive industry primarily relies on manual testing of DB SW, where a human operator visually verifies the display's accuracy and functionality. This approach is highly challenging, as bike DBs often contain hundreds of different layouts, many of which are quite similar. These layouts may correspond to the same functional group or belong to product lines with different market or language variants.


(a) DB icons expected view

(b) DB icons actual view

Figure 1.1: Comparison between DB icons expected and actual views.

Furthermore, the increased volume of vehicular state information and interactive components on the user interface complicates the task for test engineers, resulting in higher manpower demands and increased costs for the testing process. As a consequence, many OEMs have resorted to testing only random or high-priority layouts, leaving less critical functionalities out of the test plan. As SW complexity continues to grow, ensuring accuracy and comprehensive coverage becomes even more demanding, placing greater strain on testing procedures.

With the introduction of DL techniques, these challenges can be effectively addressed by integrating a CS with a HIL setup and an advanced DL based OD algorithm. In this approach, the HIL system sends a signal to the CS to capture the actual view of the DB. The CS then transmits the captured image back to the HIL, where the OD algorithm processes it by extracting all relevant assets and comparing them against reference values to determine whether the displayed view is correct. Further in this chapter, the concepts of testing, OD techniques and different CS that could be used in such a setup will be discussed in details.

## 1.2 Testing

Testing plays a vital role in the development phase, serving as an evaluation process for systems or applications to detect flaws, mistakes, or glitches, while also confirming that they align with their intended specifications and operate effectively. This practice is crucial because it guarantees that the product functions as specified, meets user requirements and maintains reliability and efficiency [2].

Testing is a continuous process that starts in the early stages of the product development and continues through the product's life cycle. Once the requirements for a product or SW are finalized, the testing process can start. The testing process is composed of a well-defined sequence of steps and is known as the SW testing life cycle [3]. The process consists of planning, designing, executing, and evaluating tests to identify and fix issues, thereby improving the quality of the product [2].

Within the SW testing life cycle, various testing methodologies are employed to ensure comprehensive product validation. One of the fundamental approaches is functional testing, which is specifically conducted to verify that the system behaves as expected and meets its defined requirements. Functional testing involves different levels of assessment, each targeting specific aspects of the SW functionality. These levels include:

- **Unit testing**: This level of testing involves testing each individual part or module on its own to ensure that it functions correctly.
- **Integration testing**: This level includes testing the integration of all the separate parts together to confirm that they work as a cohesive unit.
- **System testing**: System testing involves testing the entire system as a whole to ensure that it meets all specified requirements.
- **User acceptance testing**: This level looks at the SW from the user's perspective to confirm that it performs as they expect and meets their needs.

These testing levels are known as the testing pyramid. Figure 1.2 illustrates the test pyramid period, the higher the pyramid level, the higher the integration of the test [2].

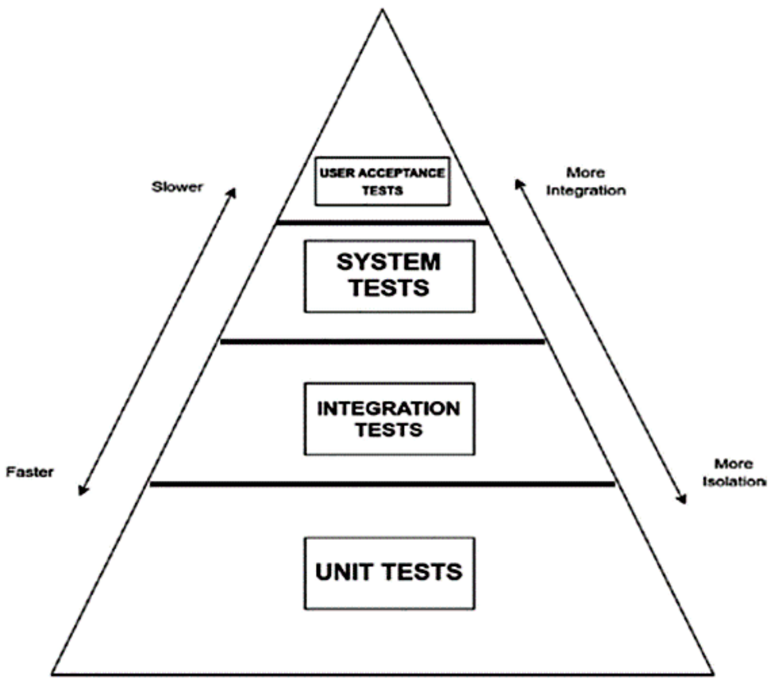


Figure 1.2: Testing pyramid consisting of 4 levels: unite tests, integration tests, system tests, user acceptance test [2]

Testing could be defined as a set of steps that are performed in order to validate or evaluate all aspects of a product. It could be done manually in a systematic approach or automated using SW scripts [4]. In the following parts of this section, manual and automated testing are explained and compared.

### 1.2.1 Manual Testing

Manual testing is a process in which test cases are executed by a human tester without the use of automated tools. The tester interacts with the system from an end-user perspective and systematically evaluating its functionalities to identify potential defects, inconsistencies and deviations from expected behavior [4].

Manual testing demands significant investment in human resources and tends to be less reliable, as the precision of tests can fluctuate due to human errors. This approach to testing is slow and significantly time-consuming.

### 1.2.2 Automated Testing

Automated testing includes specialized SW tools used to execute test cases. These tools control the test execution independently of the system being tested. The generated output is then compared to predefined expectations to validate the correct functioning of the system [2], [4].

With the growing number of functionalities and interfaces provided by the new generation of DB, automation offers a solution by executing operations consistently every time the test runs. This approach requires initial investment in the test bench hardware components and scripts development. However, it enhances the testing process efficiency and reduces the possibility of human errors [4].

### 1.2.3 Manual VS Automated Testing

The difference between automated and manual SW testing lies in their applications and benefits. Table 1.1 provides a summarized comparison of automated and manual testing. It highlights their key differences, unique advantages and applications in SW testing. Automated testing is recommended for preventing new errors in previously tested modules. On the other hand, manual testing uses the knowledge of the tester to concentrate on parts of the system thought to be more likely to have problems to excel in identifying new and unforeseen errors. These methods complement each other; automated testing allows for extensive test case coverage in a short period, while manual testing applies human insight to critically assess and target specific, potentially problematic parts of the system [4].

### 1.2.4 Testing Requirements

Testing a component or a system requires some preparation and depends on the type of testing one is planning to perform. In manual testing, the requirements are relatively minimal. For example, testing motor bike DB as a component requires a handlebar with control buttons , a CAN connector to connect to the pc that simulates the rest of the components communicating with the DB so that it does not show errors, the DB itself and a tester. Connecting all these components correctly will result in a component fully functioning and ready to test.

On the other hand, to perform the same tests in an automated approach demands sig-

Table 1.1: Comparison between Automated and Manual Testing [2], [4]

| Aspect | Automated Testing | Manual Testing |
|---|---|---|
| **Efficiency** | High efficiency due to rapid execution of tests. | Less efficient, as tests are conducted manually. |
| **Consistency** | Consistent execution, with the ability to replicate the same test scenarios multiple times. | Potential for inconsistency due to human error. |
| **Test Coverage** | Broad test coverage, can run thousands of test cases in one session. | Limited by human capacity, focuses on critical areas rather than extensive coverage. |
| **Insight** | Limited to predefined scenarios; lacks human intuition. | Benefits from human insight, allowing for the discovery of unexpected issues. |
| **Error Discovery** | Effective in identifying regressions and previously tested functionality errors. | Better at finding new and unforeseen errors through exploratory testing. |
| **Focus Areas** | Suited for repetitive, data-driven, and regression tests where consistency is key. | Focuses on high-risk areas and usability testing, relying on the tester's knowledge and experience. |

nificantly more hardware and SW than this. Using the same example of the DB testing, the basic components required would be:

- A HIL with a real time simulator to control the whole testing process.
- A CS acting as the human eye that captures the actual views on the DB.
- Mounting setup with an environment control box to always have the DB in a controlled and well viewed environment.
- An OD algorithm that will take the images captured by the CS to analyze it and extract all the assets with their locations.
- Test script that runs on the HIL to execute a sequence of test cases in an automated way.

This setup requires a significant initial investment in terms of time and money, but once it is ready, it will result in the execution of comprehensive component tests multiple times within just a few hours.

Further in this chapter the OD techniques will be discussed in details, as well as different parameters to consider when choosing a CS. This discussion will help in understanding and deciding on the appropriate CS and OD algorithm to use for capturing and analyzing the images, to contribute to the development of a complete automated test setup for DB testing.

## 1.3 Object Detection

Recognizing objects in a real-world environment using a machine presents a significant challenge. Computer vision, a specialized field within computer science, provides machines with the capability to perceive, identify, and interpret objects from images and videos. Videos are made up of a series of arranged images, called frames, displayed at a specific rate. While humans can identify objects and their positions within visual data by using their eyes to capture images and transmitting them to the brain for processing, machines rely on CS to acquire visual input and employ complex algorithms to process the data and achieve similar capabilities. This technology has been a critical area of research for decades, enabling advancements in applications such as face recognition, pedestrian tracking, security systems, vehicle identification, and autonomous driving [5]. Recently, OD has created new opportunities for test engineers to take advantage of these techniques in the field of DB testing. By integrating OD methods, engineers can develop more efficient and reliable testing processes, ultimately ensuring the delivery of the highest quality results to the end customer.

OD is a fundamental process in computer vision, it is defined as the process of integrating object classification and localization to identify and locate multiple objects within an image or video frames. Object classification assigns labels to objects by determining their categories, while object localization identifies their positions. An example of an object classification and localization (OD) is shown in figure 1.3. In a normal object recognition workflow, an image is analyzed to detect objects, assign labels corresponding to their categories, and calculate the probability for each identified class. Unique object features, such as the equal sides of a square, play a critical role in determining their categories, helping to accurately classify and distinguish between various objects within the image [5].

Choosing the right OD technique for DB testing is a vital component of this project to ensure high accuracy and reliable output. The study will explore a range of techniques, including both traditional methods and modern DL-based approaches. It will evaluate most popular detection models, analyzing their workflows and identifying their limitations. Additionally, the research will address challenges specific to this domain and investigate potential future advancements to enhance OD in DB testing. Figure 1.4 shows how the OD is divided and structured in this study, which will be further described in

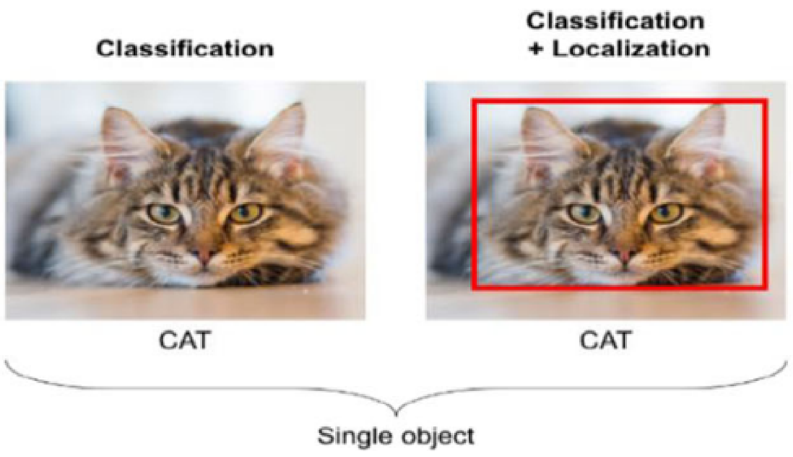


Figure 1.3: Single OD Example [5]

details in this section.

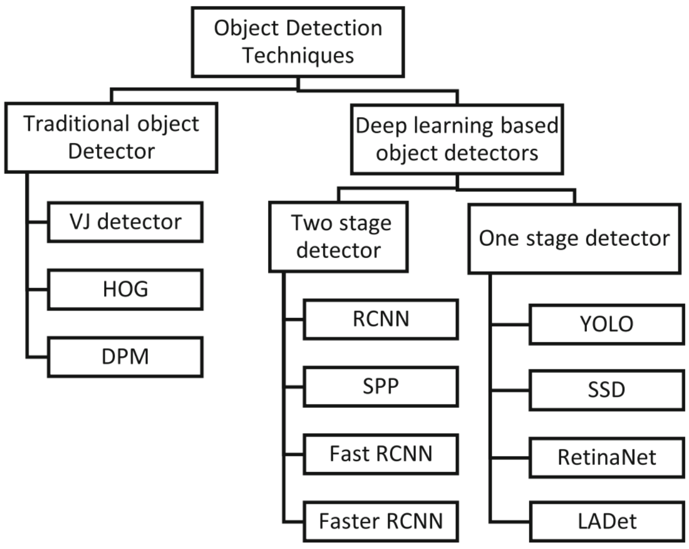


Figure 1.4: OD structure [5]

## 1.3.1 Traditional Techniques

Before modern Artificial Intelligence (AI) and DL, traditional OD methods were used to find objects in images. These methods relied on manually designed features, which are like specific rules or patterns that researchers thought would help identify objects. These features were carefully crafted to focus on important parts of the image, like edges, shapes, or textures. These important parts are known as the Region Of Interest (ROI). For example, if one wants to look for a face in a photo, the ROI might include areas with high contrast like eyes, nose, and mouth, or if one wants to detect a car in an image, the ROI might focus on the rectangular shapes of windows or wheels.

Over the years, many algorithms were developed. However, the most famous ones are:

- Viola Jones (VJ) detector which is considered to be the pioneering work that started the development of traditional OD methods in 2001 [6].
- Histograms Of Oriented Gradients (HOG) detector, a feature descriptor which was very popular for OD and image processing in 2006 [6].
- Deformable Part-Based Model (DPM) with the first introduction of bounding box regression in 2008 [6].

A detailed explanation of the three famous detectors will take place in this section. This explanation will help in developing a deeper understanding of the algorithms, their limitations and the current shift to DL techniques.

### Viola Jones Detector

The VJ detector was named after the authors of this technique “Paul Viola and Michael Jones". It is mainly used for face detection and is considered the first traditional OD technique. The detection approach is straightforward and efficient, using the sliding window technique to check all the pixels within an image and scale them to help identifying the most important parts of the image like eyes and nose as fast as possible [5]. Its operation can be divided into four main steps:

1. **Integral Image:** The input image is converted into an integral image, to perform efficient calculation of pixel intensity sums over rectangular regions. This reduces the computational cost of feature extraction for any rectangular region.
2. **Haar-like Features:** Haar-like features are used to represent patterns in the image, such as contrasts between bright and dark regions, such as darker eyes compared to brighter cheeks. These features are calculated by subtracting the sum of pixel intensities in adjacent regions.
3. **Feature Selection:** Due to the large number of potential Haar-like features, AdaBoost is used to select the most relevant characteristics. AdaBoost combines many weak classifiers into a strong classifier. It assigns higher weights to features that distinguish between faces and non-faces betters.
4. **Detection Cascade:** The last step of the VJ detector is a cascade of classifiers. It is used to discard the background region from the image and focuses the computational power on the objects itself rather than the complete image. It consists

of multiple stages where each stage filters out non-face regions. The early stages use simple classifiers to eliminate most of the image, while the later stages apply a more complex analysis to regions likely to contain faces.

While the VJ detector was a groundbreaking approach for OD it had some limitations had limitations in handling complex variations in object appearance. To overcome some of these challenges, the HOG detector was introduced.

### Histogram Of Oriented Gradients

HOG is one of the most famous feature descriptors. Feature descriptors are methods that transform raw data into a more simplified form that contains only the most important features to minimize the required computational power. HOG describes the shape and appearance of the object by analyzing the distribution of edge orientations. The HOG method calculates the gradient magnitude and orientation for each pixel in an image and partitions the image into small cells, unlike simple edge features that only detect edges without direction. The image is divided into smaller regions (localized portions), and gradients and orientations are computed for each region. A histogram is then generated for each region based on these gradients and orientations [7]. Figure 1.5 shows the comparison between the input image and the HOG image.

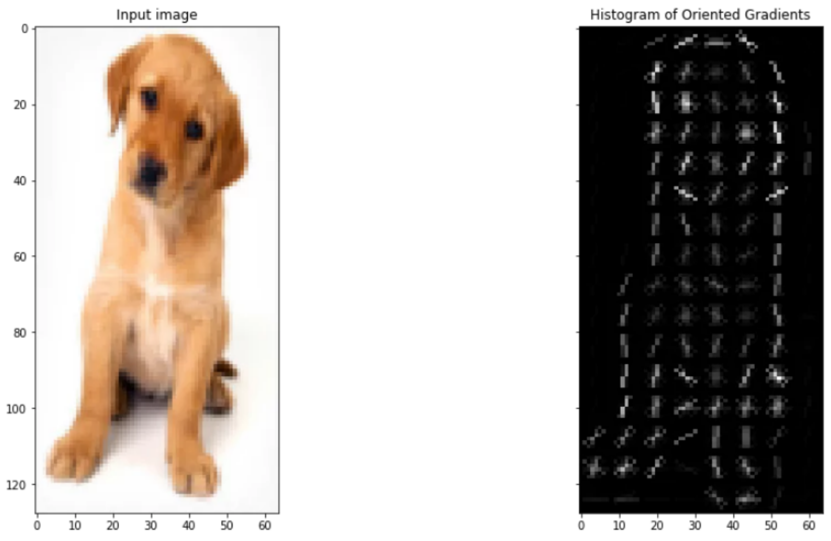


Figure 1.5: Input image and HOG image [7]

The extracted information is then passed to Machine Learning (ML) algorithms, such as Support Vector Machine (SVM), to train the classifier. To eliminate redundant bounding boxes, Non Maximum Suppression (NMS) is applied. HOG is used for detecting objects across various classes and sizes by repeatedly re-scaling the image while maintaining a fixed window size. It has been widely applied, particularly in pedestrian detection [5].

Although HOG detectors improved the robustness of the detection over VJ, they still lacked the flexibility to represent variation in objects. This limitation opened the door for DPM development to help overcome this limitation.

### Deformable Part-Based Models

DPM represent a significant leap forward in traditional OD methods. By building on the foundation of HOG detectors, DPM introduces a "divide-and-conquer" strategy. During the training or "divide" phase, the model learns how to break down objects into smaller, easily recognizable components. In the inference or "conquer" phase, these components are reassembled to predict the complete object [5].

The DPM framework is composed of three primary components: the root filter, part filters, and a spatial model. The root filter acts as the detection window, roughly outlining the entire object by applying a weighted filter to a feature vector extracted from the selected region. Meanwhile, the part filters focus on identifying smaller parts of the object, allowing the model to achieve a more detailed analysis. The spatial model then assesses the relative arrangement of the part filters in relation to the root filter. By scoring the placement of these parts, it enhances the precision of OD. This breakdown and integration of object parts make DPM a foundational tool in OD techniques [5].

### Limitations

In conclusion, most of traditional ML OD techniques depend on scanning the image and identifying the potential regions of interest that might contain the object. After that it extracts its characteristics such as HOG and other information that might be relevant. It then uses a distinct classifier to identify different types of items and determine whether objects are present within the specific subregion. This process comes with some limitations, such as the difficulty in training the classifier and feature extractor, the high computational demand, high development effort for manual feature extraction, and the failure to accurately identify the position of objects [8]. Rapidly it was realized that a more sophisticated, reliable and accurate approach is needed to fulfill the requirements of different industries.

### 1.3.2 The Shift To Deep Learning

The field of OD has seen a significant transformation, transitioning from traditional ML methods to state-of-the-art DL approaches. Unlike traditional methods that rely on hand-crafted features, modern DL-based techniques automatically learn intricate and hierarchical representations directly from the data. This capability enables more accurate, adaptable, and scalable detection systems.

DL-based detectors excel at recognizing complex patterns and finding variations within images, significantly outperforming conventional approaches, particularly in challenging tasks such as image classification and OD. By using the multi-layered feature extraction capabilities of Deep Neural Network (DNN), these systems can effectively address diverse scenarios, including variations in lighting conditions, occlusions, and object scales. This shift marks a transformative advancement in OD, providing robust and precise solutions for different industries, from autonomous vehicles to medical imaging and security systems.

What is DL, and How is it different from ML?

Both DL and ML are branches of AI, but they differ in scope and methodology. ML is a broader concept that contains a variety of algorithms capable of learning from data, including simpler techniques such as decision trees, SVMs, and ensemble models. In contrast, DL is a specialized subset of ML that uses an artificial Neural Network (NN), inspired by the structure and functioning of the human brain to process and learn complex patterns from data [9] [5].

While both ML and DL can be trained on labeled or unlabeled data, DL has an advantage when solving tasks that involve complex pattern recognition. Examples include image recognition, speech recognition, and natural language processing, where DL often outperforms traditional ML by automatically learning multi-level features without human intervention [9] [5].

At the core of DL are artificial NNs, consisting of interconnected node layers (neurons). These networks typically include an input layer, one or more hidden layers, and an output layer. Each neuron is connected to others through weighted connections and has an associated threshold. When a neuron's output exceeds its threshold, it becomes activated, passing data to the next layer; otherwise, no information is transmitted [10]. During the training process, the weights between these nodes are iteratively adjusted to minimize errors and improve performance using the activation functions. This optimization process can be achieved using approaches such as supervised learning, unsupervised learning, or reinforcement learning depending on the problem at hand [9] [5]. Figure 1.6 shows a schematic of a DNN.

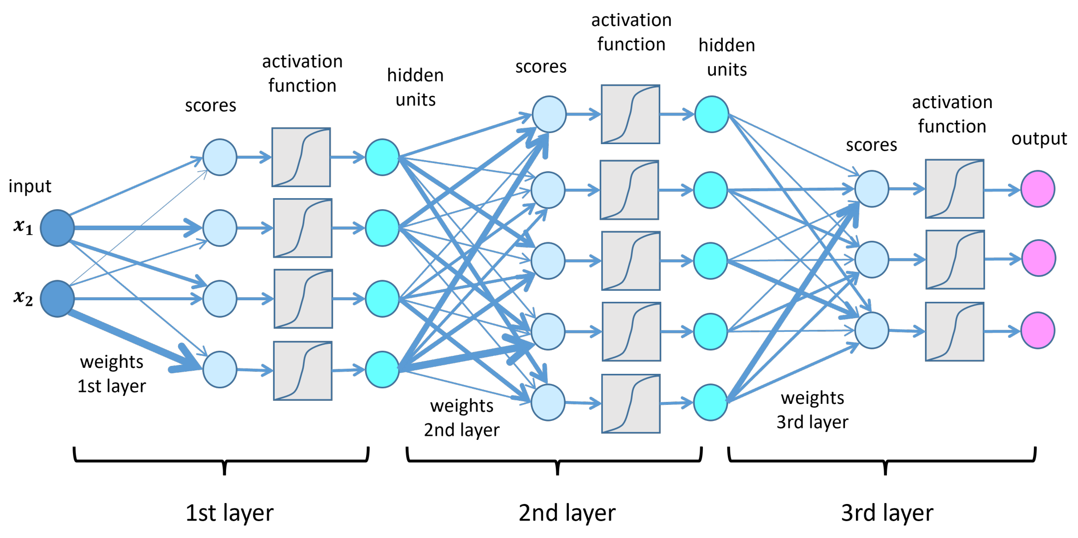


Figure 1.6: A DNN with multiple layers, illustrating the flow of data from input to output through interconnected hidden layers [11].

Once trained, a NN can efficiently analyze new unseen data and make highly accurate predictions, making it an invaluable tool for solving complex real-world problems. The ability of DL to adapt to large unstructured data sets and uncover hidden relationships within them has strengthened its role as a revolutionary advancement in AI [9] [5].

The NNs mentioned above are the simplest form of NNs and are known as feedforward NNs. However, there are different types of NNs nowadays, focusing on different tasks and data types. For example, in natural language processing and speech recognition, one can use recurrent NNs. On the other hand, when dealing with a classification problem or a computer vision task, one can depend on Convolutional Neural Network (CNN)s. CNNs offer a scalable solution for image classification and object recognition. It Uses linear algebra concepts like matrix multiplication to detect patterns within images [10].

### 1.3.3 Convolutional Neural Networks

CNNs often consists of 3 main types of layers, Convolutional Layer (CL), Pooling Layer (PL) and Fully Connected Layer (FCL). The first layer is always the CL, it could be then followed by another CL or a PL and the final layer is always an FCL. Figure 1.7 shows a schematic of a CNN architecture. As a CNN processes an image through these multiple layers, it gradually builds a more detailed understanding of the content of the image. The initial layers detect basic features like edges and colors, while deeper layers analyze more complex patterns and structures. As the data moves through the network, it begins to recognize larger shapes and meaningful components, ultimately leading to the identification of the entire object.

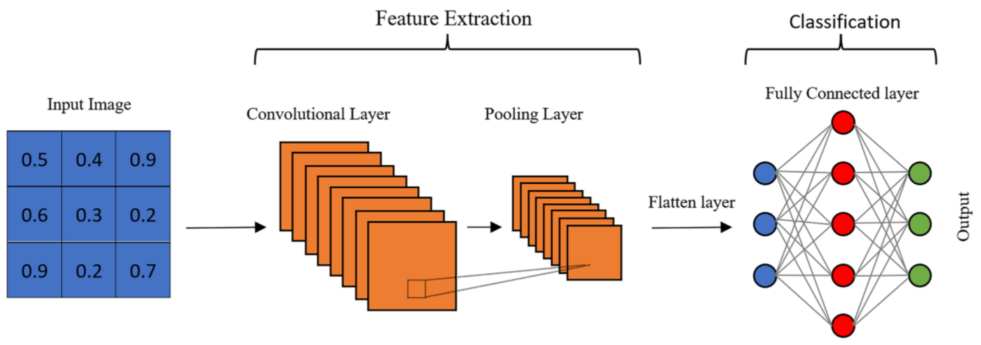


Figure 1.7: CNN architecture [12].

### Convolutional Layer

The CL is the fundamental component of a CNN, where most computations occur. It operates using three key elements: input data, filters (kernels), and feature maps. An RGB image, represented as a three dimensional matrix of pixels (height, width, and depth for RGB channels), is processed by moving a small filter across it in a process called convolution. The filter, typically a 3×3 matrix, slides over the image, computing a dot product between its weights and the pixel values, generating an activation map (feature map). Figure 1.8 explains the full process that happens within the CL. This process is controlled by three hyperparameters:

- The **number of filters**, which determines the depth of the output. Three distinct filters would result in three different feature maps, creating a depth of three [10].
- The **stride**, which defines the step size of the filter's movement. The larger the stride, the smaller the output [10].
- **zero padding**, which manages the image borders to maintain size consistency. Padding can also be **valid padding** (no padding), **same padding** (output size matches input), or **full padding** (expands the image with zeros) [10].

After convolution, a Rectified Linear Unit (ReLU) function is applied to introduce non-linearity, helping the model learn complex patterns efficiently [10].

As mentioned earlier, a CL could be followed by another CL. This setup forms a hierarchical structure where later layers analyze patterns identified by earlier ones. This

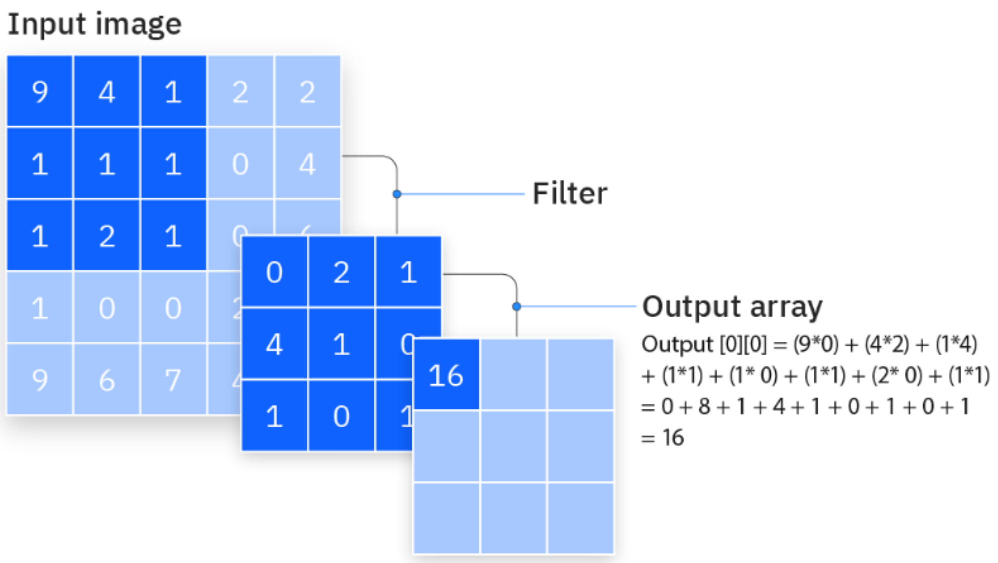


Figure 1.8: Convolutional process and dot product calculation of an input image [10].

is normally used in more complex tasks because it allows the network to break down the object into lower level features and then assemble these feature into one higher level representation [10].

### Pooling Layer

The PL benefits CNNs by lowering computational complexity, improving efficiency, reducing the risk of overfitting and helping the model generalize better to new data. It reduces the dimensionality of the data by reducing the number of parameters in the input. PL uses a sliding filter across the input, but unlike convolution, the filter does not contain weights. Instead, it applies an aggregation function to the values within its receptive field to generate a simplified output representation [10]. There are two main PL types:

- **Max pooling:** The filter selects the highest pixel value within its receptive field and passes it to the output [10].
- **Average pooling:** The filter computes the average value of the pixels in its receptive field and sends this to the output [10].

In general the max pooling is more preferred over the average pooling because it helps retain essential features. The PL comes with a downside which is loss of information. However, the benefits is much higher which makes it valuable in CNN [10].

### Fully Connected Layer

The FCL layer gets its name from its structure, where each neuron is directly linked to every neuron in the previous layer. This layer is responsible for the classification task. FCL uses the features extracted from the previous CL and PL and a softmax function, which assigns probabilities to different classes to help the model determine the most likely category for the input [10].

## 1.3.4 Deep Learning Techniques For Object Detection

DL-based OD techniques are categorized into two-stage and one-stage detectors. Two-stage detectors detect objects in two steps, often achieving state-of-the-art accuracy on benchmark datasets, but at the cost of slower inference speeds. In contrast, one-stage detectors are primarily used for real-time OD, offering faster results while maintaining satisfactory accuracy [5].

### Two Stage Detectors

Two-stage detectors, also known as multi-stage detectors, typically comprise two models. The first model identifies the ROI, while the second model classifies objects and refines their localization. These detectors are renowned for delivering state-of-the-art performance in OD, but this comes at the cost of slower processing speeds and high computational demands [5].

1. **Region Based Convolutional Neural Network (RCNN):** RCNN is a two-stage object detector that operates in three steps: region extraction, CNN feature computation, and region classification with localization. In the region extraction step, a selective search algorithm generates approximately 2000 candidate regions by grouping similar regions based on size, texture, color, and shape. Each region is then resized to a fixed size and passed through a CNN to extract features. Finally, in the region classification step, an SVM classifier assigns scores to each class, and NMS is applied to eliminate overlapping regions whose intersection over union exceeds a predefined threshold [5]. Figure 1.9 shows a schematic describing the basic steps for RCNN to detect a dog in a picture.

   This approach not only classifies objects but also provides bounding boxes around detected objects. However, RCNNs come with some limitations. Features are extracted from each proposal separately using a deep CNN, leading to redundant computations and making the training and testing processes time-consuming. Additionally, the three steps of this method are independent and operate separately, which makes it challenging to achieve a globally optimal solution. Lastly, in images with complex backgrounds, the selective search algorithm struggles to generate effective proposals because it relies solely on low-level features [5].

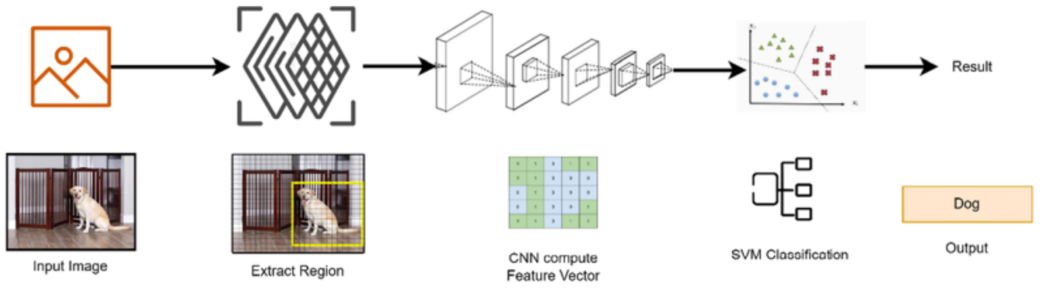


Figure 1.9: Steps involved in RCNN detector [5].

2. **Spatial Pyramid Pooling Networks (SPP-Net):** A new network was introduced in 2014 to overcome the limitations of the RCNN called SPP-Net. In RCNN, the input image is resized to a fixed size before being processed by the CNN, which can crop or distort objects, thereby reducing accuracy. SPP-Net solves this problem by removing the fixed size requirement and using an SPP layer to improve accuracy. SPP-Net processes the entire image at once to create a feature map, then divides this feature map into fixed-size grids at multiple scales as shown in figure 1.10. From each grid cell, a fixed-length feature vector is created. These feature vectors are combined and passed to soft vector machine classifiers and a bounding box regressor to detect and localize objects.

   SPP-Net achieves accuracy comparable to RCNN while being 20 times faster. However, it has its downsides. One key limitation is that it does not take full advantage of the CNN. Another drawback lies in its multi-stage training process, which can complicate implementation. Despite these limitations, SPP-Net excels at handling images with different scales and aspect ratios. It effectively preserves image details and avoids distortions that might otherwise compromise the performance of the model [5].

3. **Fast RCNN:** Further improvements were made to RCNN and SPPNet and a new architecture was introduced called Fast RCNN. Fast RCNN processes the entire input image to produce feature maps using a convolutional layer. It simplifies the design by removing multi-level pooling layers and instead uses a single grid layer. An ROI pooling layer is introduced to extract a fixed length feature vector, that acts as a simplified version of the SPP layer with just one pyramid level [5].

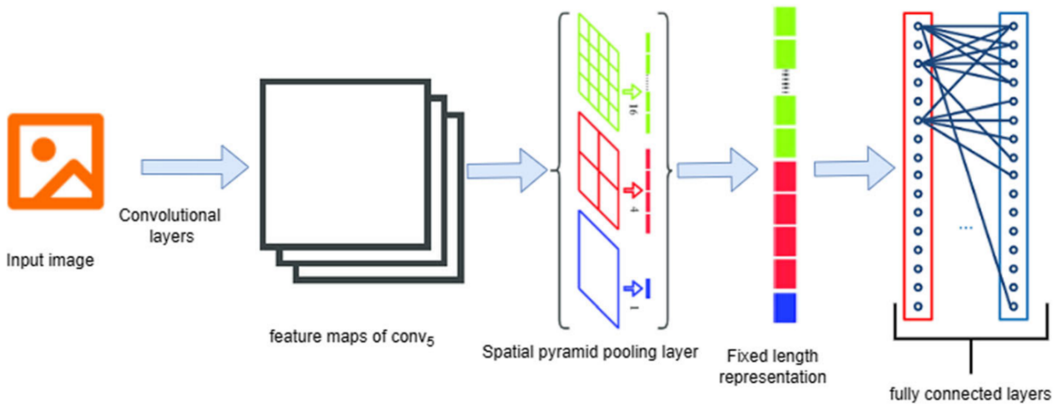


Figure 1.10: Steps involved in SPP-Net detector [5].

Each feature vector is processed through a sequence of FCL, and the final output is fed into two layers: the softmax layer and the bounding box regression layer. The softmax layer calculates probabilities for all object classes and a background class, while the bounding box regression layer generates four real values to define the bounding box for each detected object. Figure 1.11 shows the architecture used to developed a Fast RCNN [5].

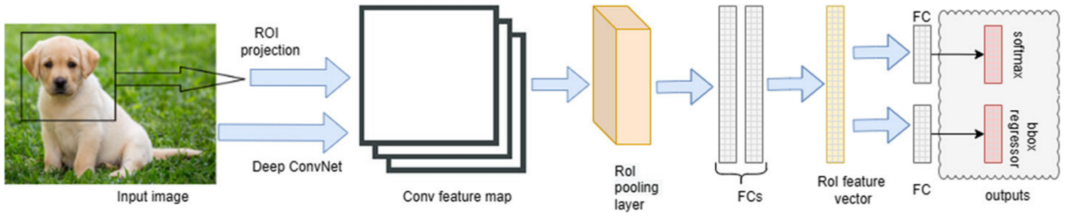


Figure 1.11: Fast RCNN architecture [5].

Fast RCNN eliminates the need for separate training for classification and bounding box regression. It was implemented using Python and C++. This method combines the strengths of RCNN and SPP-Net, offering improved accuracy and efficiency. However, it is slightly slower due to the proposal detection process. Despite this, it reduces storage requirements and enhances overall performance [5].

4. **Faster RCNN:** In 2015, a new method called Region Proposal Network (RPN) was introduced to improve the performance of Fast RCNN. RPN replaces traditional region generation methods like selective search and edge boxes. It uses a sliding window approach on the feature map to generate bounding boxes for objects, each with a confidence score. These bounding boxes, referred to as anchor boxes, are designed with a common aspect ratio. Once the proposals are generated, they are resized to a fixed size and passed through the FCL, followed by a softmax layer and a regression layer for classification and bounding box refinement. To increase non-linearity, a ReLU activation function is applied to the output of the convolution

window. This new architecture enables end-to-end training of the OD algorithm, making the process more efficient [5].

### One Stage Detector

The idea of the one stage detectors is to increase the speed of the detection and reduce the computational demand while maintaining high accuracy. This is done by remove the intermediate task to take the final output. The one stage detectors are very useful in real time detection applications that require high speed detection like autonomous systems.

1. **You Only Look Once (YOLO):** In 2015, YOLO was introduced as the first single-stage OD framework. It uses a deep CNN to process the entire input image and detect objects. The initial 20 convolutional layers are pre-trained on the ImageNet dataset, with modifications made for OD tasks. YOLO's FCL predicts both class probabilities and bounding box coordinates.

   YOLO divides the image into an S $\times$ S grid, assigning each grid cell the responsibility of detecting objects whose centers fall within it. Each cell predicts bounding boxes with confidence scores, indicating the likelihood of containing an object and the accuracy of the box.

   To improve efficiency, YOLO assigns one bounding box predictor per object based on the highest intersection over union with the ground truth. This specialization enhances recall by focusing on specific object sizes, shapes, and classes. NMS is then applied as a post-processing step to eliminate redundant bounding boxes, ensuring that only one box is retained per object [13]. Figure 1.12 explains the architecture of YOLO. YOLO kept on developing over the years and many versions are published until YOLOv11. Table 1.2 summarizes all YOLO versions and their key improvments

2. **Single Shot MultiBox Detector (SSD):** The SSD was introduced in November 2016 and marked a significant milestone in OD. It achieved at this time the highest Mean Average Precision (MAP) at a relatively high Frames Per Second (FPS) on standard datasets like PascalVOC and COCO as shown in figure 1.13. The name of the architecture reflects its key principles: "Single Shot" indicates that object localization and classification are performed in a single forward pass through the network, while "MultiBox" refers to a bounding box regression technique developed by Szegedy et al. The "Detector" aspect highlights its capability to both detect and classify objects simultaneously [15].

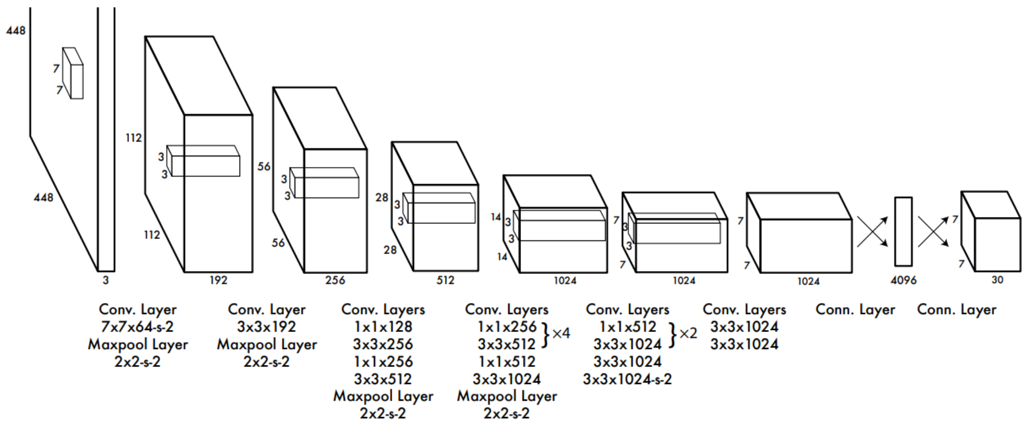


Figure 1.12: YOLO Architecture that consists of 24 convolutional layers followed by 2 FCLs [13].

| **Version** | **Key Features and Improvements** |
|---|---|
| YOLOv2 (2016) | Added batch normalization, anchor boxes, and dimension clusters to improve the original model. |
| YOLOv3 (2018) | Introduced a more efficient backbone network, multiple anchors, and SPP for enhanced performance. |
| YOLOv4 (2020) | Featured mosaic data augmentation, a new anchor-free detection head, and a new loss function. |
| YOLOv5 | Included hyperparameter optimization, experiment tracking, and automatic export to popular formats for better performance. |
| YOLOv6 (2022) | Open-sourced by Meituan and used in autonomous delivery robots. |
| YOLOv7 | Added pose estimation capabilities, including support for the COCO keypoints dataset. |
| YOLOv8 (2023) | Released by Ultralytics with improved performance, flexibility, and efficiency, supporting various vision AI tasks. |
| YOLOv9 | Introduced Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN). |
| YOLOv10 | Developed by Tsinghua University, featuring an End-to-End head that eliminates the need for NMS. |
| YOLOv11 | The latest version by Ultralytics, delivering state-of-the-art performance across multiple tasks like detection, segmentation, pose estimation, tracking, and classification. |

Table 1.2: Summary of YOLO versions and their Key Features [14]

SSD is built on the VGG-16 architecture but discards the FCLs in favor of auxiliary convolutional layers starting from conv6. This design enables the model to extract features at multiple scales while progressively reducing the input size for each subsequent layer. VGG-16 was selected as the base network due to its strong performance in high-quality image classification tasks and its effectiveness in improving results through transfer learning [15].

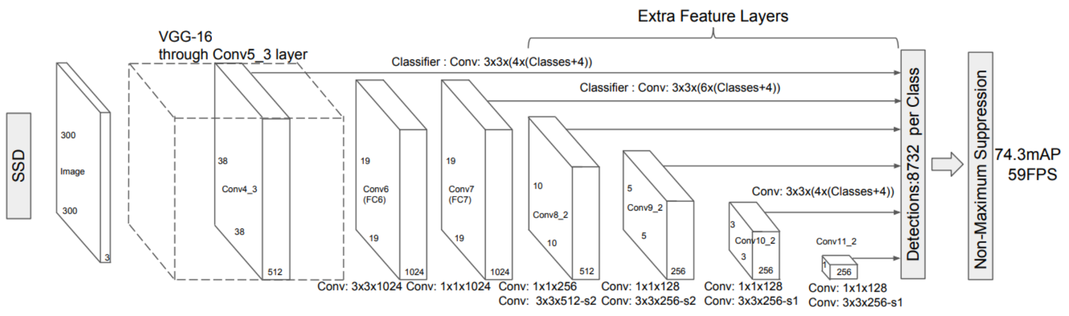


Figure 1.13: SSD Architecture that is based on VGG-16 for OD with 59 FPS [15].

3. **RetinaNet:** RatinaNet is a one-stage OD framework designed to address class imbalance during training through an innovative loss function known as focal loss. This loss function introduces a modulating factor to the standard cross-entropy loss. It focuses more on difficult examples while giving less attention to those that are already easy to classify. The RetinaNet model integrates two main components: a backbone network and two task-specific subnetworks. The backbone, typically a pre-trained CNN, generates a feature map from the input image. On top of this backbone, one subnetwork handles object classification, while the other focuses on bounding box regression [16].

   A key advantage of RetinaNet lies in its ability to handle class imbalance without relying on the cascaded sampling techniques commonly used in two-stage detectors. In traditional two-stage detectors, the proposal stage filters out most background samples, then the classification stage uses heuristics like fixed foreground-to-background ratios or online hard example mining. On the other hand, RetinaNet processes a larger set of candidate locations uniformly sampled across the image. It uses focal loss to dynamically scale the cross-entropy loss in a way that reduces its influence for well-predicted classes, allowing the model to focus on challenging cases [16].

   The network architecture is shown in figure 1.14 and includes four components: a bottom-up pathway using a ResNet backbone, a top-down pathway enhanced with feature pyramid network, a classification subnetwork, and a regression subnetwork. The feature pyramid networks allows effective feature extraction across multiple scales, facilitating the detection of objects of varying sizes. The classification subnetwork predicts the probabilities of the object class and the anchor boxes, while the regression subnetwork refines the offsets of the bounding box. This multi-scale capability makes RetinaNet particularly effective for detecting small or densely packed objects [5].

   RetinaNet's performance equals that of two-stage detectors in terms of high detection accuracy while outperforming it in fast detection speeds. Its capabilities have been demonstrated in multiple applications, including aerial and satellite imagery

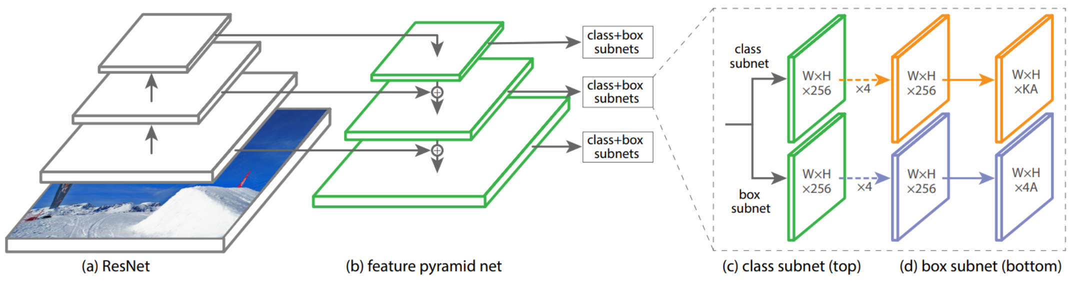


Figure 1.14: RetinaNet architecture: (a) ResNet backbone generates multi-scale feature maps, (b) Feature Pyramid Network enhances multi-scale feature extraction, (c) classification subnetwork predicts class probabilities, and (d) box subnetwork refines anchor boxes to ground-truth object boxes [5].

analysis. Furthermore, advanced variations like R4 (Refined Single-Stage Detector with Feature Recursion and Refinement) have extended its capabilities for handling complex challenges such as dense object distributions, large aspect ratios, and category imbalances [5].

4. **LADet:** LADet is a lightweight and adaptive network designed for multi-scale OD. LADet addresses challenges in OD like variations in object size through two innovative components: the Adaptive Feature Pyramid Module (AFPM) and the Light-weight Classification Function Module (LCFM). AFPM improves the integration of semantic information across different feature map levels, moving away from traditional top-down structures. LCFM optimizes anchor box usage with minimal computational overhead, using sparse convolution techniques for better efficiency. Together, these components improve detection accuracy while maintaining computational efficiency [17].

   The structure of LADet begins with a backbone network (e.g., DenseNet-169 or VGG-16) extracting multi-level feature maps. These maps are processed through the AFPM, which includes the Feature Fusion Module. Feature Fusion Module normalizes feature maps to a uniform resolution and fuses them into a unified high-resolution representation. This fusion ensures effective contribution from all scales. Complementary semantic information is added via a channel-wise attention mechanism, refining feature maps further for precise detection. The LCFM then predicts object classes and anchor box probabilities using sparse group convolutions, reducing parameter usage while maintaining performance [17]. Figure 1.15 shows an overview of the network structure.

   Experiments demonstrate that LADet is very effective on PASCAL VOC and MS COCO datasets. Its performance strikes an impressive balance between accuracy and efficiency [17].

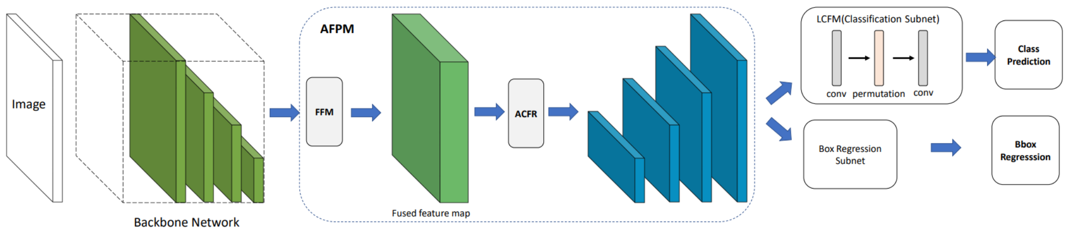


Figure 1.15: An overview of LADet model. Multi-level feature maps from the backbone network are processed by the AFPM to create pyramid feature maps. These enhanced maps are then used by the LCFM and the box regression subnet to generate classification scores and dense bounding boxes.[17].

### 1.3.5 Conclusion

The evolution of OD techniques from traditional ML methods to modern DL approaches has revolutionized the field. DL-based detectors have significantly improved the accuracy, efficiency and adaptability of OD systems. This revolution has opened new possibilities in different fields. Traditional methods like HOG and DPM have been replaced by more sophisticated models like RCNN, SSD and YOLO, which offer better performance and faster processing speeds. Moreover, they are easier to implement, available in wider range and more scalable. When merging these techniques with the right CS and the right enviromental setup, the results can be very promising. The next section will discuss the selection criteria of camera systems and how to choose the right one for a specific project.

## 1.4 Selection Criteria of Camera Systems

Choosing the right CS is a very important milestone in every computer vision project. An unsuitable system can complicate SW requirements, as it may require extensive pre-processing, lead to lost or degraded image assets, and in many cases can result in project failure. In order to avoid that, one should carefully follow a systematic approach in choosing a CS. This process requires a careful evaluation of key features to ensure that the hardware is optimally aligned with the project's specific objectives.

Key technical considerations in CS selection include the scanning method, resolution, Dynamic Range (DR), sensor size, frame rate, shutter type, and Field Of View (FOV). Additional factors, such as lens type, communication protocol, mounting techniques, environmental conditions, and budget requirements, should also be evaluated to improve the robustness of the system and its compatibility with the demands of the project.

### 1.4.1 Scanning Method

The camera's scanning or imaging method is a primary factor in determining the appropriate camera type for a given application. The most commonly used scanning methods are line scan and area scan.

#### Line Scan Cameras

Line scan cameras operate with a single row of pixels arranged on a long, narrow sensor. As the target object moves along the narrow axis of the sensor, the camera captures data one row of pixels at a time, constructing the complete image through sequential SW processing. This method requires careful synchronization between the sensor scan time and the movement of the object to produce a stable and accurate image output. The continuous image-building capability makes line scan cameras particularly well suited for high-speed applications such as print inspection and document scanning, where maintaining image quality over fast moving objects is critical [18].

A primary advantage of line scan cameras is their ability to capture continuous images without being limited by a fixed vertical resolution, resulting in significantly higher resolution than area scan cameras. Additionally, these cameras are optimized for high-speed processing environments, as they enable rapid image acquisition and efficient data transfer. Since the pixel readout rate is faster than the exposure time, line scan cameras can initiate a new capture while the previous image is still transferring, maximizing their efficiency in fast-paced settings. This feature makes them ideal for applications that demand rapid, uninterrupted image acquisition [18].

However, line scan cameras also come with limitations. The need for precise synchronization between the sensor's capture timing and the moving object's speed is crucial to maintaining image stability, which can be challenging to achieve in dynamic environments. Furthermore, line scan cameras are limited in their ability to capture static images or video of moving objects, as they rely on the movement of the object to construct the image. The installation process is often complex and costly, requiring specialized setups that can increase the overall expense of a project. Despite these limitations, line scan cameras remain invaluable in high-resolution, high-speed applications where traditional imaging methods may fall short [18].

**Area Scan Cameras**

On the other hand, area scan cameras have a matrix of pixels that captures the data of the image in a single exposure. The camera structure of an area scan camera typically consist of an array sensor, such as a charge coupled device or complementary metal oxide semiconductor known as CMOS, where each pixel represents part of the entire image captured in a single exposure [18].

Area scan cameras are a more cost-effective choice since they are simpler to use and easier to install than line scan cameras. Their versatility extends to tasks requiring strobe lighting for moving objects and allows for real-time video display on monitors, a feature which is very much appreciated when real time video processing is required. Unlike line scan cameras, which must capture multiple rows of data to assemble a full image, area scan cameras can capture a defined area in a single exposure which results in saving time and simplifying the imaging process. Additionally, area scan cameras offer the flexibility to divide a single image into multiple ROI to allow specific OD within different areas of the frame simultaneously. However, it still comes with the limitation in its FOV, unlike the line scan camera, which has a wide FOV, the area scan camera is restricted in this aspect. Figure 1.16 illustrates a comparison between area scan and line scan cameras in terms of their working principles.

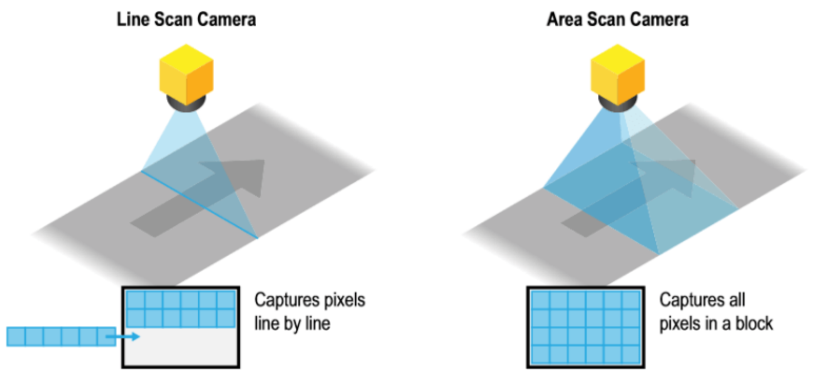


Figure 1.16: Comparison of area scan and line scan cameras [19].

### 1.4.2 Resolution

Camera resolution is a key parameter to be considered when working with DL projects. It refers to the number of pixels used to represent an image. Image resolution directly influences the quality of data input into a DL model. Higher resolution images provide more detailed information, which can improve the model's performance for tasks such as OD, image segmentation, and classification. For example, in medical imaging or satellite data analysis, higher resolution can capture fine details that are important for diagnosis or analysis. However, as the resolution increases, so does the computational cost in terms of memory and processing time, making it important to find a balance between image quality and efficiency.

Several studies have investigated the impact of image resolution on CNN performance across various fields, including medical imaging. For instance, Thambawita et al. [20] conducted an experimental study using the HyperKvasir dataset, which consists of 10,662 endoscopic images covering 23 different gastrointestinal conditions. Their research focused on how changes in image resolution influence CNN performance, particularly in the context of endoscopic image classification. The study focused on image resolutions ranging from 32×32 to 512×512 pixels and evaluated the performance of two CNN architectures: ResNet-152 and DenseNet-161. Figure 1.17 shows the differences in image quality for different image resolutions.

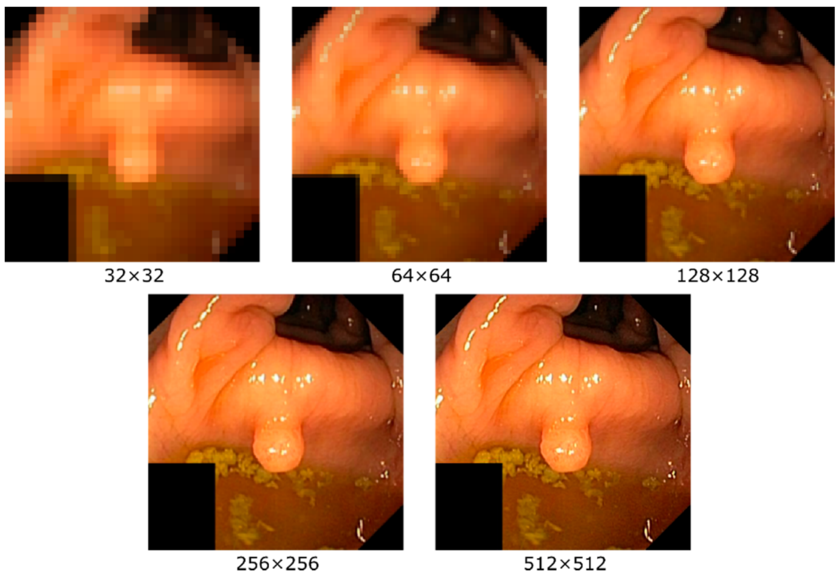


Figure 1.17: Example of images that were used in the study to show the differences in level of details that are observed from different resolutions. Note that for this figure all pictures are rescaled to the same size to show quality differences [20].

The study [20] found that higher resolution images consistently led to improved classification accuracy and precision. Specifically, the use of 512×512 pixel images resulted in the highest performance metrics. The research highlighted that high-resolution images better preserved fine details, such as surface textures and subtle features of lesions, which are essential for accurate classification in medical contexts. On the other hand, lowering the resolution caused a notable decline in performance, particularly in detecting subtle abnormalities like Barrett's esophagus.

It was also highlighted in a study by Barkat et al. [21], capturing images at 42 MP resolution, combined with High Dynamic Range (HDR) techniques, significantly improves the clarity of visual data, especially in low-light conditions. This improvement enables more accurate detection and classification of events like avalanches using DL algorithms. Furthermore, the high-resolution images provide detailed information that allows for the identification of subtle surface deformations, which are essential for detecting early signs of avalanches. The study highlights that such critical details would be missed with lower resolution images, stressing the importance of maintaining high resolution in optical monitoring systems to ensure both accuracy and reliability in DL-based detection.

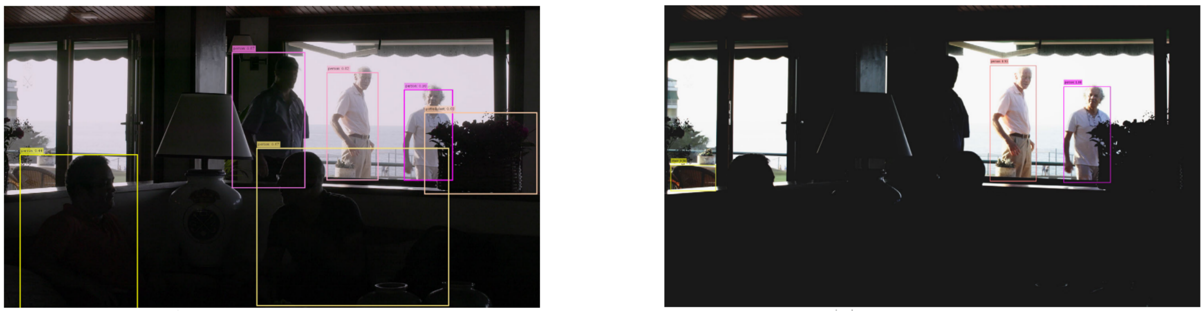
One of the primary challenges highlighted in the study by Thambawita et al. [20] was the significant increase in computational cost associated with processing higher-resolution images. The use of larger images in the training of CNN requires more memory and processing power. As a result, it becomes necessary to reduce the batch size to fit these higher-resolution images within the limited capacity of the available GPU memory. Even with the extra computational demands, the research showed that using higher-resolution images led to better performance, which made the trade-offs worthwhile.

In the context of DL-based text and object recognition for motorbike DBs, similar findings can be drawn from the studies. For example, a key issue is recognizing text with low contrast against the background or in small fonts, making it difficult for standard recognition systems. Using higher resolution camera images can improve clarity, allowing models to better detect and process this information. However, this solution must be balanced with the computational power required, as higher resolution images demand more resources. To ensure feasibility in practical applications, it is important to optimize both accuracy and processing efficiency in these systems.

### 1.4.3 Dynamic Range

DR is defined as the ratio between the highest and lowest light intensities a system can handle without losing information. In digital imaging, this range often reflects how well a camera can adapt to bright and dim elements in the same scene. Mukherjee et al. describe HDR as an advanced imaging technique designed to capture and process a wide spectrum of visible lighting conditions, HDR imaging encompasses a broad range of luminance values, spanning approximately 13 orders of magnitude, from $10^{-4}$ to $10^{8}$ cd/m$^2$ [22]. Direct sunlight can have luminance values up to approximately $1.6 \times 10^{9}$ cd/m$^2$ [23].This large range of brightness levels is why HDR is used to capture both very dark and very bright areas in a scene accurately. To capture and represent this range, HDR scene information is often stored in high precision formats, using 32 or 16 bits per pixel per channel, in file types like `.hdr` or `.exr`. floating point formats. This range is closely aligned with the human eye's ability to perceive various light levels. In contrast, Standard Dynamic Range (SDR) imaging compresses scene details into a more limited range, around 3 orders of magnitude, which can restrict visibility in extreme lighting. HDR, on the other hand, captures luminance from very dark shadows to intensely bright highlights, preserving critical details essential for precise OD in complex lighting environments [22].

DR is considered one of the most important parameters when choosing a CS for DL projects especially in the field of OD. Many studies discussed the impact of DR on detecting objects in challenging lighting conditions. for instance, Mukherjee et al. presented for the first time HDR imaging to OD. The study provided a methodology to test and compare the performance of HDR and SDR trained models under difficult lighting conditions. The authors used for this study Faster RCNN and SSD models and their findings revealed that models trained on HDR data perform 10–12% better in accuracy than those trained on SDR data [22]. Figure 1.18 show the comparison presented in the study between HDR detection (Figure 1.18a) and SDR Detection (Figure 1.18b). It is clear that HDR trained detectors successfully detected all assets of the picture, while SDR failed to detect 4 of the assets.



(a) HDR Detection (b) SDR Detection

Figure 1.18: Example image showing the comparison results between HDR and SDR detection [22].

Using HDR imaging in DL offers several benefits, such as improved model accuracy and stronger resilience to varying real-world lighting challenges. HDR provides richer visual information, allowing models to train on a wider range of lighting conditions, which enhances their ability to generalize and reduces the likelihood of errors in unfamiliar settings. This makes HDR valuable in applications where accuracy and reliability are important. However, HDR also brings certain challenges. The higher bit-depth and extended range of HDR images demand more storage and computational power for both training and inference, which can lead to longer processing times and higher costs during deployment.

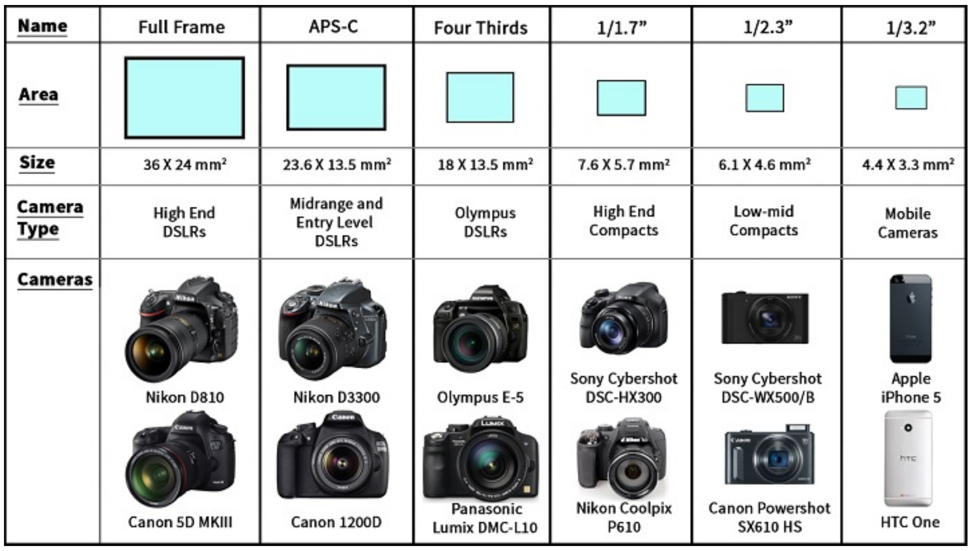
To overcome this limitation, one can use tone mapping operators to map the HDR content to SDR content without losing the assets in the original scene. This is done using a tonecompression curve; it works by changing the 16/32 bit floating point to 8 bit channels. Resulting in a pixel value ranging from 0 to 255. [22].

### 1.4.4 Sensor Size

The term sensor size in CSs refers to the physical dimensions of the imaging sensor, which captures light and converts it into electrical signals to create digital images. Figure 1.19 Shows how sensor size changes between multiple devices. But is it correct to say that bigger sensor size means better image with more details? Answering this question is not so simple and depending on the configuration of the photodiods in the sensor. Photodiods are widely known as pixels, it consists of a semiconductor p-n junction and its fundamental role is to absorb light and convert it to electrical signal. Different manufacturers use different configuration of pixels in camera sensor to satisfy different needs.

To explain this in simple terms, one can link the performance of the sensor to its size, when the sensor is big in size like the full frame sensor.

Generally, larger sensors area are able to capture more light, resulting in images with more detail, less noise, and enhanced DR. Recent research highlights the importance of sensor size in determining image quality, especially in low-light settings or high-contrast scenes. This makes sensor size a critical factor in applications such as OD, image classification, and scene recognition within DL, where high-quality images can significantly impact model performance.



| **Name** | Full Frame | APS-C | Four Thirds | 1/1.7” | 1/2.3” | 1/3.2” |
|---|---|---|---|---|---|---|
| **Area** | ▭ | ▭ | ▭ | ▭ | ▭ | ▭ |
| **Size** | 36 X 24 $mm^2$ | 23.6 X 13.5 $mm^2$ | 18 X 13.5 $mm^2$ | 7.6 X 5.7 $mm^2$ | 6.1 X 4.6 $mm^2$ | 4.4 X 3.3 $mm^2$ |
| **Camera Type** | High End DSLRs | Midrange and Entry Level DSLRs | Olympus DSLRs | High End Compacts | Low-mid Compacts | Mobile Cameras |
| **Cameras** | Nikon D810<br>Canon 5D MKIII | Nikon D3300<br>Canon 1200D | Olympus E-5<br>Panasonic Lumix DMC-L10 | Sony Cybershot DSC-HX300<br>Nikon Coolpix P610 | Sony Cybershot DSC-WX500/B<br>Canon Powershot SX610 HS | Apple iPhone 5<br>HTC One |

Figure 1.19: Use case of different sensor sizes and their used application [24].

Another crucial factor to consider when selecting a sensor size is the required resolution, as resolution and sensor size are closely related. In simple terms, resolution refers to the number of pixels in an image as explained earlier. If the sensor is small but the resolution requirement is high, each pixel will occupy only a tiny space on the sensor, receiving minimal light. This limited light per pixel generally leads to lower image quality and increased noise. Conversely, with a larger sensor, each pixel has more room and can capture a greater share of light, which enhances overall image quality by reducing noise and improving detail.

In a study [25] investigates how specific camera parameters, including sensor pixel size, influence the generalization of CNNs in OD tasks. The authors used synthetic data generated with the ISETAuto soft prototyping tool, which simulates camera images under various conditions to control parameters like pixel size, bit-depth, and exposure. In this experiment, the effect of changing sensor's characteristics on CNN ability to generalize across datasets was evaluated, including synthetic and real-world images. The study found that pixel size significantly impacts model performance, especially in generalization across varied environments. One can directly relate pixel size with sensor size because it is the amount of physical dimension required per pixel, and that is directly proportion to the size of the sensor for a given resolution.

In terms of methodology, The authors created synthetic images with varying pixel sizes (from 1.5 µm to 6 µm) and trained a CNN on these images. They then evaluated the model's ability to generalize to other datasets, including KITTI, BDD, and CityScape, which are standard datasets in autonomous driving research. The results demonstrated that smaller pixel sizes generally led to better generalization performance, likely due to the higher spatial resolution afforded by smaller pixels, which allows the model to capture finer details essential for OD. However, as pixel size increased, the model's ability to generalize decreased, resulting in poorer performance on datasets with varying lighting and scene conditions.

This study illustrates the critical role of sensor size (specifically pixel size) in determining the quality of data for DL applications. Smaller pixel sizes, associated with higher spatial resolution, enable CNNs to learn more detailed features, which is essential for tasks requiring precision, such as OD in varying environments. However, larger pixel sizes, while potentially more light-sensitive, may reduce resolution and, consequently, model performance in diverse scenes. These findings suggest that optimizing sensor size for specific DL tasks is essential, as it directly impacts the accuracy, robustness, and generalization capabilities of AI models, particularly in complex real-world environments like autonomous driving.

### 1.4.5 Frame Rate

Frame rate, expressed in FPS, plays a crucial role in the performance of OD models in computer vision. Higher frame rates can provide more frequent image updates, enabling the system to detect fast-moving objects with greater precision and maintain smoother visual tracking. This allows for improved responsiveness and accuracy in dynamic environments. In a study to optimize OD in autonomous vehicles using gray scale computer vision models [26], the authors argued that increasing frame rate enhances OD accuracy by providing higher temporal resolution, particularly in dynamic environments like autonomous vehicle navigation.

The authors claimed that high frame rates play a crucial role in real-time OD by proving that it minimizes the distance a vehicle travels between captured frames, therefor enhancing decision-making speed. For example, at a velocity of 80 km/h, a camera operating at 60 FPS corresponds to 37 cm of travel per frame. Doubling the frame rate to 120 FPS reduces this distance to 18.5 cm per frame, allowing earlier and more accurate detection and response to objects. This improvement is particularly significant in high-speed and complex scenarios where timely and precise decisions are needed. Moreover, higher frame rates improve the detection of fast-moving or distant objects by providing a more continuous and detailed input stream for the detection model.

In an application such as DB testing where objects and texts are stationary for the current generation, it's not necessary to consider such a parameter when choosing a CS. However, it is expected to have high more animations and graphics in the future generations of DB, that is why it could be a good investment to include such a parameter in the decision making process if the resources allow.

# Bibliography

[1] A. X. Qun Pang et al., “Deep learning based layout recognition approach for hmi software validation,” in *2024 IEEE Conference on Artificial Intelligence (CAI)*, 2024, pp. 429–437. DOI: 10.1109/CAI59869.2024.00085.

[2] P. Leloudas, *Introduction to Software Testing*. Berkeley, CA: Apress, 2023.

[3] Vedpal and N. Chauhan, “Role of machine learning in software testing,” in *2021 5th International Conference on Information Systems and Computer Networks (ISCON)*, 2021, pp. 1–5. DOI: 10.1109/ISCON52037.2021.9702427.

[4] S. Shiwangi, R. Gadgil, and A. Chudgor, “Automated testing of mobile applications using scripting technique: A study on appium,” *International Journal of Current Engineering and Technology (IJCET)*, vol. 4, no. 5, pp. 3627–3630, 2014.

[5] J. Kaur and W. Singh, “Tools, techniques, datasets and application areas for object detection in an image: A review,” *Multimedia Tools and Applications*, vol. 81, no. 27, pp. 38 297–38 351, 2022, ISSN: 1573-7721. DOI: 10.1007/s11042-022-13153-y. [Online]. Available: https://doi.org/10.1007/s11042-022-13153-y.

[6] G. Boesch. “Object detection: The definitive 2025 guide.” Accessed: 2025-02-24. [Online]. Available: https://viso.ai/deep-learning/object-detection/.

[7] A. Vidhya. “Feature engineering for images – introduction to hog feature descriptor.” Accessed: 2024-12-16. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/.

[8] M. Hameed and Z. Khalaf, “A survey study in object detection: A comprehensive analysis of traditional and state-of-the-art approaches,” *Basrah Researches Sciences*, vol. 50, p. 16, Jun. 2024. DOI: 10.56714/bjrs.50.1.5.

[9] Google Cloud. “What is deep learning?” Accessed: 2024-06-17. [Online]. Available: https://cloud.google.com/discover/what-is-deep-learning.

[10] IBM. “What are convolutional neural networks?” Accessed: 2025-02-10. [Online]. Available: https://www.ibm.com/think/topics/convolutional-neural-networks.

[11] Lamarr Institute. “Deep neural networks.” Accessed: 2024-06-17. [Online]. Available: https://lamarr-institute.org/blog/deep-neural-networks/.

[12] A. Alsaleh and C. Perkgoz, “A space and time efficient convolutional neural network for age group estimation from facial images,” *PeerJ Computer Science*, vol. 9, e1395, May 2023. DOI: 10.7717/peerj-cs.1395.

[13] R. Kundu. “Yolo: Algorithm for object detection explained [+examples].” Accessed: 2025-01-13. [Online]. Available: https://www.v7labs.com/blog/yolo-object-detection.

[14] Ultralytics. “Yolo: A brief history.” Accessed: 2025-01-13. [Online]. Available: https://docs.ultralytics.com/%5C#yolo-a-brief-history.

[15] W. Liu et al., “Ssd: Single shot multibox detector,” in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[16] P. with Code. “Retinanet.” Accessed: 2025-01-14. [Online]. Available: https://paperswithcode.com/method/retinanet.

[17] J. Zhou, Y. Tian, W. Li, R. Wang, Z. Luan, and D. Qian, “Ladet: A light-weight and adaptive network for multi-scale object detection,” in *Proceedings of The Eleventh Asian Conference on Machine Learning*, W. S. Lee and T. Suzuki, Eds., ser. Proceedings of Machine Learning Research, vol. 101, PMLR, Nov. 2019, pp. 912–923. [Online]. Available: https://proceedings.mlr.press/v101/zhou19a.html.

[18] Q. Technologies. “Area scan vs line scan cameras: Which one should you choose?” Accessed: 2024-10-29. [Online]. Available: https://qualitastech.com/image-acquisition/area-scan-vs-line-scan-cameras/.

[19] L. I. Systems. “Glossary of print inspection terms.” Accessed: 2024-10-29. [Online]. Available: https://lakeimage.com/glossary-of-print-inspection-terms/.

[20] V. Thambawita, I. Strümke, S. A. Hicks, P. Halvorsen, S. Parasa, and M. A. Riegler, “Impact of image resolution on deep learning performance in endoscopy image classification: An experimental study using a large dataset of endoscopic images,” *Diagnostics*, vol. 11, no. 12, p. 2183, 2021. DOI: 10.3390/diagnostics11122183.

[21] I. Barkat et al., “Optical monitoring of avalanche release zones in bessans, haute maurienne valley, france, with a remote and energy self-sufficient camera system,” in *Proceedings of the International Snow Science Workshop*, Tromsø, Norway, 2024, pp. 1041–1048.

[22] R. Mukherjee, M. Bessa, P. Melo-Pinto, and A. Chalmers, “Object detection under challenging lighting conditions using high dynamic range imagery,” *IEEE Access*, vol. 9, pp. 77 771–77 783, 2021. DOI: `10.1109/ACCESS.2021.3082293`.

[23] LampHQ. “Illuminance vs luminance: What’s the difference?” Accessed: 2024-11-12. [Online]. Available: `https://lamphq.com/illuminance-vs-luminance/`.

[24] A. Larmon. “Cell phones versus a camera: Sensor size matters.” Accessed: 2024-11-19. [Online]. Available: `https://larmonu.larmonstudios.com/cell-phones-versus-a-camera-sensor-size-matters/`.

[25] Z. Liu, T. Lian, J. Farrell, and B. Wandell, “Neural network generalization: The impact of camera parameters,” *arXiv preprint arXiv:1912.03604*, 2019, Accessed: 2024-11-12. [Online]. Available: `https://arxiv.org/abs/1912.03604`.

[26] D. Liu and A. Moch, “Optimizing object detection in autonomous vehicles using grayscale computer vision models,” TRITA-EECS-EX-2024:342, Degree Project in Computer Science and Engineering, First Cycle, KTH Royal Institute of Technology, 2024.