



UNIVERSITY OF THESSALY  
SCHOOL OF ENGINEERING  
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Object Detection via Transformers in Manufacturing  
Industry for Human Safety**

Diploma Thesis

**Matthaios Dimitrios Tzimas**

**Supervisor:** Dimitrios Rafailidis

June 2023





UNIVERSITY OF THESSALY  
SCHOOL OF ENGINEERING  
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Object Detection via Transformers in Manufacturing  
Industry for Human Safety**

Diploma Thesis

**Matthaios Dimitrios Tzimas**

**Supervisor:** Dimitrios Rafailidis

June 2023

iii





ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Ανίχνευση Αντικειμένων μέσω Μετασχηματιστών στη  
Μεταποιητική Βιομηχανία για την Ανθρώπινη Ασφάλεια**

**Διπλωματική Εργασία**

**Ματθαίος Δημήτριος Τζήμας**

**Επιβλέπων:** Ραφαηλίδης Δημήτριος

Ιούνιος 2023



Approved by the Examination Committee:

Supervisor **Dimitrios Rafailidis**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Dimitrios Katsaros**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Gerasimos Potamianos**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly





# Acknowledgements

I would like to express my sincere gratitude to my supervisor, associate professor Dimitrios Rafailidis, for his guidance. Also, I would like to thank my family and my best friends for their unwavering encouragement and support.



## **DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS**

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Matthaios Dimitrios Tzimas

## Diploma Thesis

# Object Detection via Transformers in Manufacturing Industry for Human Safety

**Matthaios Dimitrios Tzimas**

## Abstract

The safety of workers in the manufacturing industry is a critical issue. Computer vision can increase the safety of workers through object detection. Object detection has the ability to detect multiple objects in an image, such as humans, helmets. Detectors like YOLO-V7 and Faster R-CNN can handle a small dataset like ours, compared to detection transformers (DETR), which require a large amount of data. In order to improve the low performance of DETR, we use a pre-trained ResNET50 which is the backbone of DETR, the rest of the network is initialized based on the authors implementation, and a pre-trained detection transformer. We then separately fine-tune them on our dataset. The first one is pre-trained with self-supervised learning on the imageNet dataset. The results based on the AP50 metric are similar to those of YOLO-V7 and Faster-RCNN. The second one is pre-trained with supervised learning on the COCO dataset, and it outperformed the competing algorithms we investigated.

### **Keywords:**

Machine learning, Deep learning, Computer vision, Detection transformers.

## Διπλωματική Εργασία

### Ανίχνευση Αντικειμένων μέσω Μετασχηματιστών στη Μεταποιητική Βιομηχανία για την Ανθρώπινη Ασφάλεια

Ματθαίος Δημήτριος Τζήμας

## Περίληψη

Η ασφάλεια των εργαζομένων στη μεταποιητική βιομηχανία είναι ένα κρίσιμο ζήτημα. Η υπολογιστική όραση μπορεί να αυξήσει την ασφάλεια των εργαζομένων μέσω της ανίχνευσης αντικειμένων. Η ανίχνευση αντικειμένων έχει την ικανότητα να ανιχνεύει πολλαπλά αντικείμενα σε μια εικόνα, όπως ανθρώπους, κράνη. Ανιχνευτές όπως οι YOLO-V7 και Faster R-CNN μπορούν να χειριστούν ένα μικρό σύνολο δεδομένων όπως το δικό μας, σε σύγκριση με τους μετασχηματιστές ανίχνευσης (DETR), οι οποίοι απαιτούν μεγάλο όγκο δεδομένων. Προκειμένου να βελτιώσουμε τη χαμηλή απόδοση του DETR, χρησιμοποιούμε ένα προ-εκπαιδευμένο ResNET50 το οποίο αποτελεί τον κορμό του DETR, το υπόλοιπο δίκτυο αρχικοποιείται με βάση την υλοποίηση των συγγραφέων, και έναν προ-εκπαιδευμένο μετασχηματιστή ανίχνευσης. Στη συνέχεια, τους ρυθμίζουμε ξεχωριστά στο σύνολο δεδομένων μας. Ο πρώτος είναι προεκπαιδευμένος με αυτοεπιβλεπόμενη μάθηση στο σύνολο δεδομένων imageNet. Τα αποτελέσματα με βάση τη μετρική AP50 είναι παρόμοια με εκείνα των YOLO-V7 και Faster-RCNN. Ο δεύτερος είναι προεκπαιδευμένος με επιβλεπόμενη μάθηση στο σύνολο δεδομένων COCO, και ξεπέρασε τις επιδόσεις των αντίπαλων αλγορίθμων που μελετήσαμε.

### Λέξεις-κλειδιά:

Μηχανική μάθηση, Βαθιά μάθηση, Όραση υπολογιστών, Μετασχηματιστές ανίχνευσης.



# Table of contents

|   |             |
|---|-------------|
| <b>Acknowledgements</b>                                   | <b>ix</b>   |
| <b>Abstract</b>   | <b>xii</b>  |
| <b>Περίληψη</b>   | <b>xiii</b> |
| <b>Table of contents</b>                                  | <b>xv</b>   |
| <b>List of figures</b>                                    | <b>xvii</b> |
| <b>List of tables</b>                                     | <b>xix</b>  |
| <b>Abbreviations</b>                                      | <b>xxi</b>  |
| <b>1 Introduction</b>                                     | <b>1</b>    |
| <b>2 Related Work</b>                                     | <b>3</b>    |
| 2.1 Machine and Deep learning . . . . .                   | 3           |
| 2.1.1 Artificial Neural Networks . . . . .                | 5           |
| 2.1.2 Activation Functions . . . . .                      | 6           |
| 2.1.3 Convolutional Neural Networks . . . . .             | 12          |
| 2.2 Computer Vision . . . . .                             | 15          |
| 2.3 Object Detection . . . . .                            | 18          |
| 2.3.1 Two-stage Detectors . . . . .                       | 19          |
| 2.3.2 One-Stage Detectors . . . . .                       | 20          |
| 2.4 Self Supervised Learning in Computer Vision . . . . . | 22          |

---

|          |                              |           |
|----------|------------------------------|-----------|
| <b>3</b> | <b>Examined Models</b>       | <b>25</b> |
| 3.1      | Faster-RCNN . . . . .        | 25        |
| 3.2      | YOLO-V7 . . . . .            | 27        |
| 3.3      | DETR . . . . .               | 29        |
| 3.4      | DINO . . . . .               | 31        |
| <b>4</b> | <b>Experiments</b>           | <b>33</b> |
| 4.1      | Experiment Setup . . . . .   | 33        |
| 4.2      | Dataset . . . . .            | 33        |
| 4.3      | Settings . . . . .           | 34        |
| 4.3.1    | Metrics . . . . .            | 34        |
| 4.3.2    | Hyper-parameters . . . . .   | 37        |
| 4.4      | Results . . . . .            | 37        |
| 4.4.1    | Faster-RCNN Models . . . . . | 37        |
| 4.4.2    | YOLO-V7 Models . . . . .     | 41        |
| 4.4.3    | DETR . . . . .               | 54        |
| 4.5      | Summary . . . . .            | 56        |
| <b>5</b> | <b>Conclusions</b>           | <b>59</b> |
| 5.1      | Future work . . . . .        | 59        |
|          | <b>Bibliography</b>          | <b>61</b> |



# List of figures

|      |   |    |
|------|---|----|
| 2.1  | Supervised learning workflow[1] . . . . .                                       | 4  |
| 2.2  | Visualization of Perceptron [2] . . . . .                                       | 6  |
| 2.3  | ANNs [3] . . . . .  | 7  |
| 2.4  | Binary Step Function [4] . . . . .  | 7  |
| 2.5  | Linear activation function [4] . . . . .  | 8  |
| 2.6  | Sigmoid activation function [4] . . . . .                                       | 9  |
| 2.7  | Derivative of sigmoid [4] . . . . .   | 9  |
| 2.8  | Tanh activation function [5] . . . . .  | 10 |
| 2.9  | ReLU activation function [5] . . . . .  | 11 |
| 2.10 | Convolution process after one operation [6] . . . . .                           | 13 |
| 2.11 | Left:max pooling , right : average pooling [6] . . . . .                        | 13 |
| 2.12 | Residual block [7] . . . . .  | 15 |
| 2.13 | 3D Object Reconstruction from a Single Color Image [8] . . . . .                | 16 |
| 2.14 | Visualization of image classification problem [9] . . . . .                     | 17 |
| 2.15 | Object detection of cats and dogs [10] . . . . .                                | 17 |
| 2.16 | Semantic Segmentation [11] . . . . .  | 18 |
| 3.1  | Left : Faster-RCNN network[12] , Right : Region Proposal Network [13] . . . . . | 26 |
| 3.2  | ELAN computational block [14] . . . . .   | 27 |
| 3.3  | E-ELAN computational block [14] . . . . .                                       | 28 |
| 3.4  | scaling width and depth on a concatenation-based model [15] . . . . .           | 28 |
| 3.5  | DETR pipeline [16] . . . . .  | 29 |
| 3.6  | Architecture of DETR [16] . . . . .   | 30 |
| 3.7  | DINO student and teacher networks [17] . . . . .                                | 31 |
| 4.1  | Left : train-valid-test split ,Right : Instances per class . . . . .            | 34 |

---

|      |  |    |
|------|--|----|
| 4.2  | Some examples of dataset instances . . . . .   | 34 |
| 4.3  | Intersection Over Union [18] . . . . .   | 35 |
| 4.4  | Precision-Recall curve of each class with threshold $a = 0.5$ (from our experiments) . . . . . | 36 |
| 4.5  | Faster-RCNN R50-FPN training stats . . . . .   | 38 |
| 4.6  | Faster-RCNN R101-FPN training stats . . . . .  | 39 |
| 4.7  | Faster-RCNN X101-FPN training stats . . . . .  | 40 |
| 4.8  | YOLO-V7 1 training stats . . . . .   | 41 |
| 4.9  | YOLO-V7 2 training stats . . . . .   | 42 |
| 4.10 | YOLO-V 73 training stats . . . . .   | 43 |
| 4.11 | YOLO-V7 4 training stats . . . . .   | 44 |
| 4.12 | YOLO-V7 5 training stats . . . . .   | 45 |
| 4.13 | YOLO-V7 6 training stats . . . . .   | 46 |
| 4.14 | YOLO-V7X 7 training stats . . . . .  | 47 |
| 4.15 | YOLO-V7X 8 training stats . . . . .  | 48 |
| 4.16 | YOLO-V7X 9 training stats . . . . .  | 49 |
| 4.17 | YOLO-V7X 10 training stats . . . . .   | 50 |
| 4.18 | YOLO-V7X 11 training stats . . . . .   | 51 |
| 4.19 | YOLO-V7W 12 training stats . . . . .   | 52 |
| 4.20 | YOLO-V7W 13 training stats . . . . .   | 53 |
| 4.21 | Training stats of DETR without pre-train . . . . .   | 54 |
| 4.22 | Training stats of DETR (COCO) . . . . .  | 54 |
| 4.23 | Training stats of DETR(DINO) . . . . .   | 55 |

# List of tables

|      |   |    |
|------|---|----|
| 4.1  | Faster-RCNN R50 FPN results and hyper-parameters . . . . .  | 38 |
| 4.2  | Faster-RCNN R101 FPN results and hyper-parameters . . . . . | 39 |
| 4.3  | Faster-RCNN X101 FPN results and hyper-parameters . . . . . | 40 |
| 4.4  | YOLO-v7 tiny 1 . . . . .                                    | 41 |
| 4.5  | YOLO-v7 tiny 2 . . . . .                                    | 42 |
| 4.6  | YOLO-v7 3 . . . . .   | 43 |
| 4.7  | YOLO-v7 4 . . . . .   | 44 |
| 4.8  | YOLO-v7 5 . . . . .   | 45 |
| 4.9  | YOLO-v7 6 . . . . .   | 46 |
| 4.10 | YOLO-V7X 7 . . . . .  | 47 |
| 4.11 | YOLO-v7X 8 . . . . .  | 48 |
| 4.12 | YOLO-v7X 9 . . . . .  | 49 |
| 4.13 | YOLO-v7X 10 . . . . .                                       | 50 |
| 4.14 | YOLO-v7X 11 . . . . .                                       | 51 |
| 4.15 | YOLO-V7W 12 . . . . .                                       | 52 |
| 4.16 | YOLO-V7W 13 . . . . .                                       | 53 |
| 4.17 | DETR without pre-train . . . . .                            | 54 |
| 4.18 | Pre-train DETR on COCO . . . . .                            | 54 |
| 4.19 | Pre-trained DETR with DINO . . . . .                        | 55 |
| 4.20 | Summary of all experiments . . . . .                        | 56 |
| 4.21 | DETR (COCO) hyper-parameter tuning . . . . .                | 56 |



# Abbreviations

|      |  |
|------|--|
| CNN  | Convolutional Neural Network             |
| IoU  | Intersection over Union                  |
| ANN  | Artificial Neural Network                |
| PPE  | Personal protective equipment            |
| DETR | DEtection TRansformer                    |
| EMA  | Exponential Moving Average               |
| FC   | Fully Connected                          |
| mAP  | Mean Average Precision                   |
| AP   | Average Precision                        |
| AP50 | Average precision with 0.5 IoU threshold |
| LR   | Learning Rate                            |
| RPN  | Region Proposal Network                  |



# Chapter 1

## Introduction

Worker safety is a significant concern in the manufacturing industry, with numerous head injuries and fatalities reported annually. According to [19], over 50,000 head injuries occur each year, resulting in more than 1,000 deaths. Surprisingly, 84% of these incidents happen to workers who are not wearing any head protection. To address this problem, computer vision has emerged as a promising solution that can detect whether workers are wearing necessary protective equipment using cameras. Among various subfields of computer vision, object detection has been considered the most suitable for this problem. Detection transformers (DETR) [16] are powerful detectors, and we use them in order to detect humans, helmets, and vests in images from our dataset [20]. There are numerous approaches to object detection, and many of them utilize the detectors from the YOLO family [21, 22], including YOLO-v5 and YOLO-v3 [23]. To investigate the most up-to-date YOLO algorithm, we will explore YOLO-v7 [15] and compare it with Faster R-CNN [24], a detector known for its high performance. Interestingly, both of these methods can perform well on small datasets like ours, in contrast to DETR, which typically requires large amounts of data [25]. To overcome this limitation, we use a pre-trained DETR [16] and a pre-trained DETR backbone [17], and we fine-tune them on our dataset. The first one is pre-trained with supervised learning on the COCO [26] dataset. The COCO dataset consists of 330,000 images with over 1.5 million object instances. However, creating a dataset like COCO can be an arduous and costly process. As a result, we also use a pre-trained DETR backbone, which is trained on ImageNet [27] using the self-supervised learning algorithm DINO. DINO can train a network without the need for annotations, significantly lowering the cost of creating a large dataset.

This contribution is summarized as follows :

1. We investigate the object detection algorithms YOLOv7, Faster-RCNN and DETR.
2. We train them on a dataset that is able to detect the necessary Person Protection Equipment (PPE).
3. For each of the investigated detectors we utilize a version of them that was pre-trained with supervised learning and fine tune them on the PPE dataset.
4. For the DETR algorithm we also utilize a backbone that was pre-trained with self supervised learning.
5. We evaluate the models, and we find out that the pre-trained with supervised learning DETR has the best results on the PPE dataset.

The rest of the thesis is structured as follows: Chapter 2 provides a comprehensive review of deep learning techniques, computer vision methods, Convolutional Neural Networks (CNNs), and architectures for object detection. This chapter aims to establish a solid foundation of knowledge and understanding in these areas. In Chapter 3, an in-depth analysis of the examined models is presented. Chapter 4 focuses on presenting the results obtained from the experiments, accompanied by a detailed description of the experiment settings. Finally, Chapter 5 provides a summary and conclusion of the experiments, highlighting the main findings and contributions of this thesis. Additionally, this chapter offers suggestions and ideas for future work.



# Chapter 2

## Related Work

### 2.1 Machine and Deep learning

Machine learning is a type of artificial intelligence that educates computers to learn from their experiences. Rather than depending on a pre-established equation as a model, machine learning algorithms use computational methods to gather information directly from data where are manually extracted by data scientists. As more examples become accessible for learning, the algorithms progressively enhance their performance. Deep learning, on the other hand, as a subcategory of machine learning is also teaching computers to learn by examples but with the difference that they take raw data as input [28]. Both of the above fields share common learning methods with the most popular being supervised learning and unsupervised learning.

#### 1. Supervised learning

The objective of supervised learning given a set of input-target pairs, is to learn a function that maps input to target output. The function for a given unknown input makes predictions based on the patterns that are been discovered in the mapping process also called training. Supervised learning is typically used in scenarios where there is a clear outcome or target variable that needs to be predicted or classified. For instance, in an image recognition problem, the input data may be an image, and the output data could be a label indicating what is present in the image, such as a cat or a dog.

The labeled dataset is divided into training and testing. The training dataset is been used to find the best possible function which is usually represented by a mathematical model

that can accurately predict the target data based on the input data. After training we evaluate the model on test dataset by measuring the distance between the predicted and the target output. If the distance is low, it means that the model has learned to generalize well from the training data and can make accurate predictions for new, unseen data [29]. Supervised learning uses classification and regression techniques to develop a machine learning model. Classification when the output variable is categorical or discrete and regression when it is numerical [28]. Also is widely used in various fields such as image and speech recognition, natural language processing, fraud detection, and many others. Some popular algorithms used for supervised learning include decision trees, support vector machines, k-nearest neighbors, random forests [29].

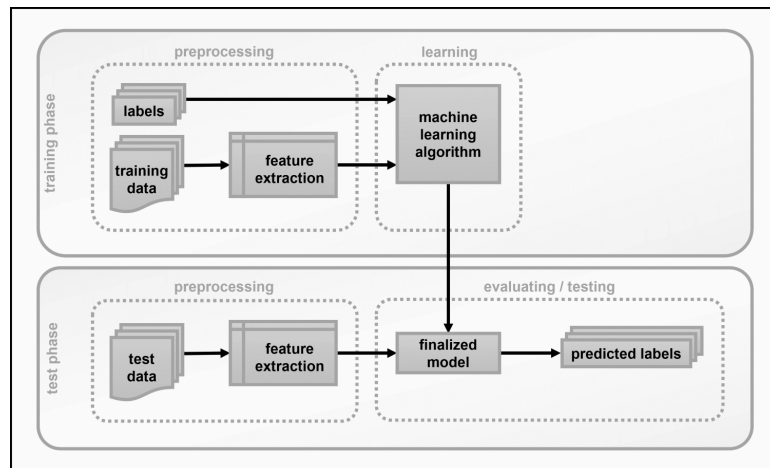


Figure 2.1: Supervised learning workflow[1]

## 2. Unsupervised learning

Unsupervised learning algorithms operate on an unlabeled dataset, allowing them to freely extract features and identify patterns without any external supervision. Clustering and feature reduction are two commonly used techniques in unsupervised learning. Clustering involves grouping similar data points together into clusters based on their similarities and differences, whereas feature reduction involves reducing the number of features or variables in the data while retaining as much information as possible [29].

## 3. Semisupervised Learning

Semi-supervised learning can be described as a combination of the previously mentioned methods, as it requires both labeled and unlabeled data for the training process.

In this method, predictions on the unlabeled data based on the labeled data are used as targets for training. This process reduces the time and cost needed for annotation while achieving better results than unsupervised models [30].

#### 4. Reinforcement learning

Reinforcement learning is considered the closest attempt at modeling the human learning experience because it learns through interaction with its environment. This technique is based on rewarding desired actions and punishing undesired ones, with the ultimate goal of maximizing the rewards. As the algorithm interacts with its environment, it learns to make decisions that lead to positive outcomes and avoid actions that lead to negative outcomes, allowing it to continually improve its performance over time [30].

### 2.1.1 Artificial Neural Networks

Artificial neural networks (ANNs), also referred to as neural networks, constitute a subset of machine learning and are a crucial component of deep learning algorithms. ANNs are primarily employed for universal function approximation in numerical paradigms because of their exceptional features, such as self-learning, adaptivity, fault tolerance, nonlinearity, and proficiency in mapping input to output. Due to these qualities, ANNs are better equipped to tackle intricate and uncomplicated issues in various domains like agriculture, manufacturing, and medical science than conventional mathematical algorithms. Thus, ANNs are gaining popularity in solving real-world problems that are difficult to solve using conventional methods [31].

The concept of neural networks is inspired by the structure and function of the human brain. Our brains are made up of neurons that form intricate connections and communicate through electrical signals to process information [32]. Similarly, artificial neural networks consist of artificial neurons or nodes that collaborate to solve complex problems. These nodes are software modules, and the neural networks themselves are algorithms or software programs that utilize computing systems to perform mathematical calculations [32]. To facilitate comprehension of this concept, a visual representation and an explanation of the simple ANN called Perceptron is presented. In a perceptron, the input values  $x_i, i \in 1, 2, \dots, n$  are multiplied by their corresponding weights  $w_i, i \in 0, 1, \dots, n$  and added together to produce

a weighted sum. This weighted sum is then passed through an activation function, which ensures that the output is mapped to the desired range [2].

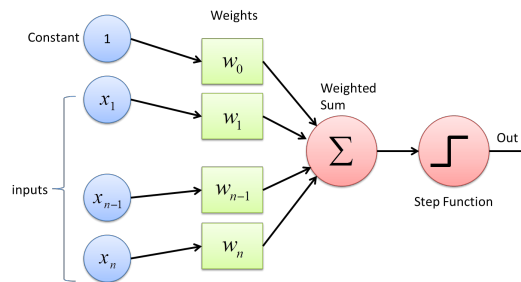


Figure 2.2: Visualization of Perceptron [2]

- Input,  $\mathbf{x}_i, i \in 1, 2, \dots, n$  and  $x_0 = 1$
- Weights,  $\mathbf{w}_i, i \in 0, 1, \dots, n$
- Weighted sum,  $S = \sum_{i=1}^n \mathbf{w}_i \cdot \mathbf{x}_i + w_0 \cdot 1$
- Activation Function  $f$ , in our case Step Function 2.1.2
- Output  $Y = f(S)$

In order to address complex problems, deep learning employ a layered architecture consisting of interconnected nodes. Typically, a basic neural network is composed of three layers, namely the input, hidden, and output layers. The input layer serves as the initial stage where the input data is introduced to the network. Subsequently, the hidden layer(s) follows and applies a series of nonlinear transformations to the input data. Finally, the output layer produces the ultimate output of the network based on the input data and the learned features from the hidden layer(s). This layered structure provides the neural network with the ability to model complex relationships between inputs and outputs, resulting in powerful and flexible machine learning models [32].

### 2.1.2 Activation Functions

The inclusion of activation functions is crucial to ensure the efficacy of artificial neural networks, as they enable the transformation of the dot product of inputs with their corresponding weights [4]. The utilization of a linear function or the absence of an activation function altogether will result in the neural network functioning like a linear regression model, and

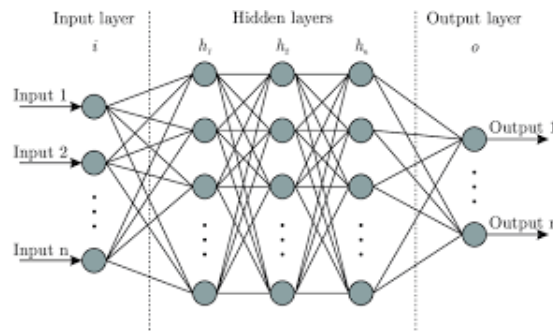


Figure 2.3: ANNs [3]

will face challenges in detecting complex patterns in the data. It is, therefore, imperative to introduce non-linearity in neural networks through the adoption of differentiable activation functions [33]. Activation functions can be classified into three groups:

### 1. Binary step function

The binary step function is typically utilized for binary classification tasks, where the output of this activation function is constrained to two possible values based on a defined threshold. The resulting output value governs the activation of the corresponding neuron in the network and the gradient is always equal to zero [33].

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \quad (2.1)$$

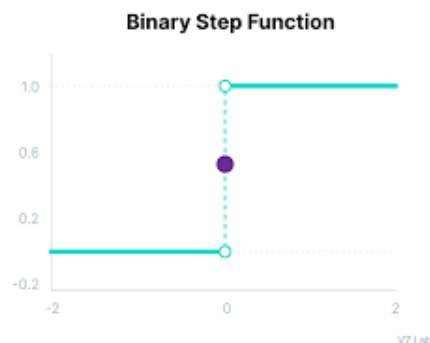


Figure 2.4: Binary Step Function [4]

### 2. Linear activation function

If we solely rely on this function, it will result in the collapse of the neural network into one layer irrespective of the number of original hidden layers [4]. Also one characteristic of the linear activation function is that its gradient remains constant, regardless of the input value it receives which makes difficult the optimization [33].

$$f(x) = x \quad (2.2)$$

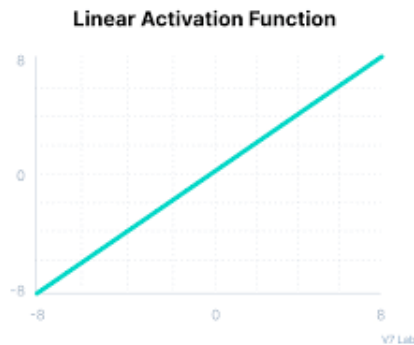


Figure 2.5: Linear activation function [4]

### 3. Non linear Activation Function

- **Sigmoid function**

The values of sigmoid function  $f$  have a range between 0 and 1. As the input  $x \rightarrow \inf$ ,  $f \rightarrow 0$  and when  $x \rightarrow \sup$ ,  $f \rightarrow 1$ . Because of the range of the output values the sigmoid function is used in neural network where the output is a probability  $p \in [0, 1]$ . Also the no symmetry around zero or in other words the absence of the negative sign outputs makes the neural network more unstable and difficult to train [4].

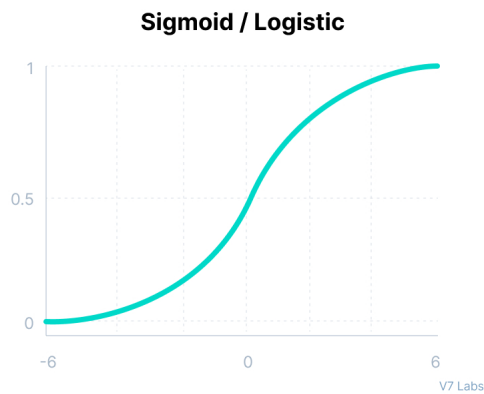


Figure 2.6: Sigmoid activation function [4]

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

The derivative of the sigmoid is :

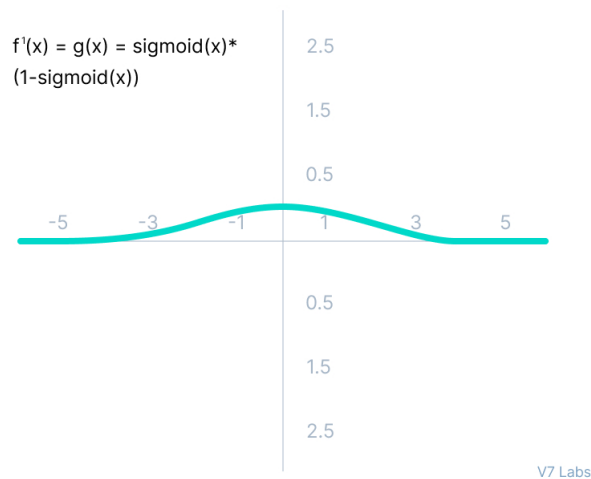


Figure 2.7: Derivative of sigmoid [4]

$$f'(x) = f(x)(1 - f(x)) \quad (2.4)$$

The facts  $f' \in (0, 0.25)$  and that is  $f'$  is only significant for  $x \in (-3, 3)$  makes the training difficult because the network suffers from the Vanishing gradient problem [4, 34].

- **Tanh function**

The implementation of the hyperbolic tangent (tanh) function is particularly common within hidden layers due to its symmetry around zero [4]. This feature allows for both positive and negative values to be transmitted to the next layer. In addition to its zero symmetry, the tanh function also has an advantage over the sigmoid function in terms of derivative range [33]. Specifically, the range of the derivative for the tanh function spans from zero to one which is four times greater compared to sigmoid but it also suffers from the vanishing gradient problem [35].

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.5)$$

The figure 2.8 [5] illustrates the graphical representation of the equation 2.5 [4].

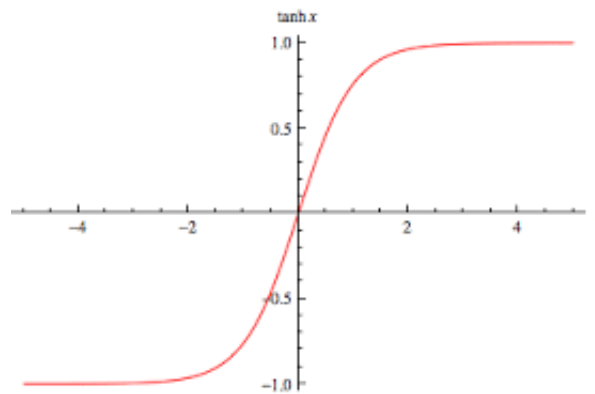


Figure 2.8: Tanh activation function [5]

#### 4. ReLU

Back-propagation is possible with this computationally efficient activation function, which speeds up neural network training. The equation is 2.6 and the graphical representation of the formula 2.9 [4]:

$$f(x) = \max(0, x) \quad (2.6)$$



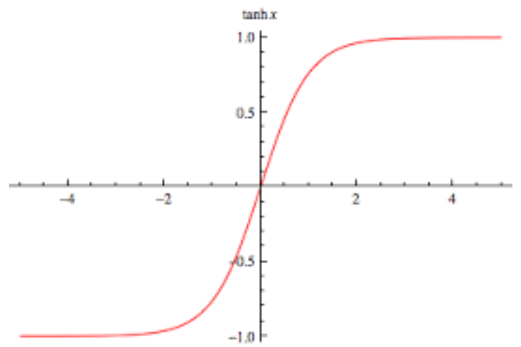


Figure 2.9: ReLU activation function [5]

ReLU thus only activates neurons with input greater than zero and the derivative is 2.7[36] :

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \quad (2.7)$$

To deal with the the zero output values and zero gradient for  $x < 0$ , a modifications of the Relu function exists which is called parametric ReLu 2.8 with derivative 2.9 [4, 36].

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ a \cdot x, & \text{if } x < 0 \end{cases} \quad (2.8)$$

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ a, & \text{if } x < 0 \end{cases} \quad (2.9)$$

## 5. Softmax

Softmax is very important activation function in a variety of deep learning applications. Also is mainly used in the last layer of the neural network in order to transform the K output vector into a K probability distribution with sum equal to 1 [37].

$$f(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (2.10)$$

### 2.1.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are responsible for the surge of computer vision in the era of deep learning, as they eliminate the need for manual feature extraction, which was previously used in traditional computer vision methods. CNNs utilize linear algebra to identify patterns in data, especially in images, despite the computational cost associated with this process [38]. "Neocognitron" [39], proposed by Kunihiko Fukushima in 1980, was an early deep learning model designed for image recognition. It was based on the structure of the visual cortex in the brain and used a series of convolutional layers to extract features from images. However, at the time of its development, there were limited optimization techniques available for supervised learning, which made it difficult to train deep neural networks effectively. As a result, Neocognitron was not able to achieve the same level of performance as modern deep learning models [40]. Since then, significant advances in optimization techniques, such as backpropagation and stochastic gradient descent [41], have enabled the development of more powerful deep learning models that can achieve state-of-the-art performance on a wide range of image classification tasks [42].

The difference between CNNs and other deep neural networks is that the hidden layers of CNNs are mostly composed of convolutional, pooling and fully connected layers [43, 44]. The purpose of convolutional layers in deep learning is to extract relevant features from the input data [45]. This is achieved by applying convolution between the input data and the kernel which is a matrix with learnable weights. When we are stacking many kernels together the three dimensional matrix that is produced is called filter. Each element of the output matrix is the dot product of the kernel with the corresponding area of the input [38]. Non-linearity is incorporated by feeding the convolutional output to an activation function, resulting in a feature map output [46]. The size of a feature map in a convolutional layer is determined by the kernel size, stride, and padding. The kernel size refers to the dimensions of the matrix used for convolution, while stride specifies the amount of input area skipped before computing the next dot product. Padding is the additional space added around the input matrix before convolution. These hyper-parameters significantly impact the performance of the convolutional operation, and their careful consideration is essential for the effective design and optimization of convolutional neural networks [47].

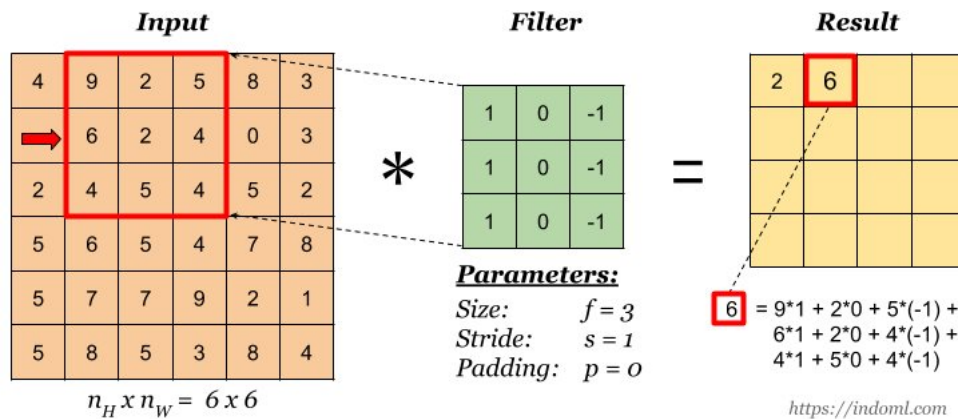


Figure 2.10: Convolution process after one operation [6]

Convolutional neural networks use sampling reduction methods, such as the pooling layer, to reduce the size of the input. Unlike the kernels of convolutional networks, the pooling layer uses the max and average pooling methods, whose kernels have no learning weights. Max pooling selects the highest value of the corresponding input region, while average pooling calculates the average of the corresponding values within the input region [46]. Pooling methods have hyper-parameters the kernel size, stride and the type (average or max) [6]. Typically, a  $2 \times 2$  kernel size with a  $2 \times 2$  stride is used to halve the input, although kernel sizes larger than 3 can reduce the accuracy of the model. Such techniques are vital for improving computational efficiency [47].

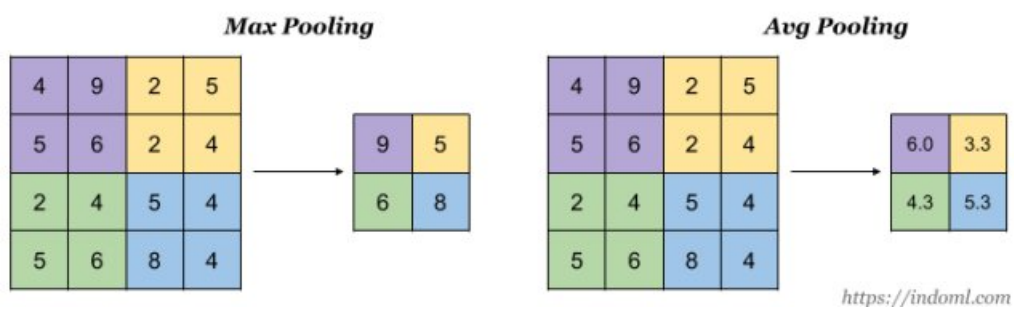


Figure 2.11: Left: max pooling , right : average pooling [6]

In the fully connected layer each neuron is connected to all inputs and is the last layer of a CNN. We use one or more fully connected layers to analyze the high level features that are in the last three dimensional feature map. For this reason we convert this three dimensional matrix into a vector (1D matrix) which is the input of the first fully connected layer [46]. The activation functions used in this layer vary many times depending on the problem we have to solve [38].

We will analyze in chronologically ascending order some of the neural networks that opened the way to the level at which computer vision is today:

### 1. Lenet 5

LeNet-5 was proposed by LeCun et al. (1998) [41]. This convolution neural network designed for handwritten digit recognition outperform older methods because of the application of backpropagation across the multilayer network. It has seven layers, three convolution, two pooling and two fully connected. The convolutions and fully connected layers have around sixty thousand learnable parameters and their weights are updated via stochastic gradient descent [41].

### 2. AlexNet

AlexNet was proposed by Krizhevsky et al. in 2012 [48]. This deep convolution neural network was trained on ImageNet which is a dataset with over one million images and one thousand classes. The five convolution layers, sometimes combined with a max pooling layer and the three fully connected builds a model with sixty million parameters [48]. The authors in order to accelerate the training process is using the Relu non linear activation function and multiple graphics processing units. Compared to other networks the pooling layer has squared kernels with size equal to three and stride two which causing overlapping pooling regions [48]. This overlap reduce the error. To deal with the problem of over-fitting they apply data augmentation and dropout to the model [49]. The AlexNet is also using stochastic gradient descent as the optimization algorithm [48].

### 3. VGG-16

VGG-16 was proposed by Simonyan and Zisserman in 2014 [50]. The authors choose convolution kernel size equal to three as it is the smallest possible size to capture spatial information. This small kernels with stride one are producing larger feature maps allowing the network to get deeper [50]. Relu serves as the activation function in the hidden layers, while the pooling layer uses max pooling with kernel size 2 and stride 2 to prevent overlapping regions [50]. The architect has 19 layers with weights, 16 convolution, 3 fully connected and it was trained with mini batch gradient descent with momentum [50].

#### 4. ResNet

Residual Network was proposed by He et al. in 2015 [7]. The authors' primary objective is to address both the degradation problem and vanishing or exploding gradients. This occurrence occurs when the number of hidden layers increases [7]. Due to the difficulty of determining the complex and nonlinear relationship between the input  $X$  and the underlying mapping  $H(X)$ , the authors proposed the block 2.12 to map the function  $f(X) = H(X) - X$ , so that  $H(X) = F(X) + X$  [7]. As shown in 2.12, they accomplish this by adding a connection, which skips layers. For  $F(x)$  to be added to  $X$ , the matrices must have the same dimension. Therefore, they employ convolutions with a kernel size of 3 and a stride of 1, which yields an output of the same size as the input [7]. If they do not utilize these convolutions, they multiply  $X$  by a matrix with learnable or constant parameters so that the output dimension matches the input dimension. Adding many blocks the neural network is able to get deeper without increasing the error [7]. ResNet was trained with stochastic gradient descent, without dropout and ReLU as the main activation function [7].

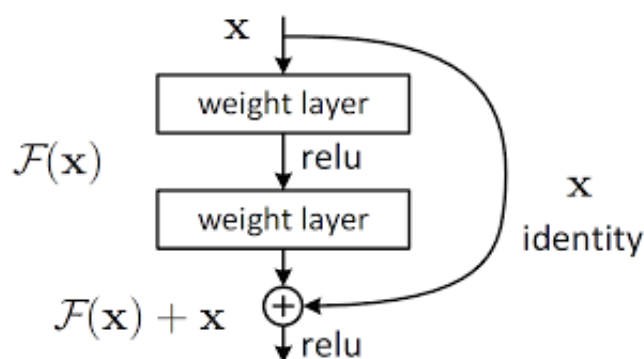


Figure 2.12: Residual block [7]

## 2.2 Computer Vision

The field of artificial intelligence known as computer vision aims to extract and elucidate information acquired from images and video. While the human brain possesses an advantage in this regard due to years of evolution, the ability of computers, aided by algorithms, to analyze vast quantities of images per second has the potential to surpass human capabilities [51].

Below we will list the most common computer vision techniques used in the construction

and manufacturing industry [52].

### 1. 3D reconstruction

In order to address this particular issue, it is necessary to acquire a collection of images that can be used to reconstruct a three-dimensional view. The reason for this is that a single image alone does not contain sufficient information about depth, which is lost in the process of projecting the three-dimensional view onto a two-dimensional image. While there is a method for accomplishing this, it is not widely known due to the significant number of requirements involved. These requirements include the Calibration Problem, Matching Problem, Reconstruction Problem, and the Density of the Reconstruction [53].

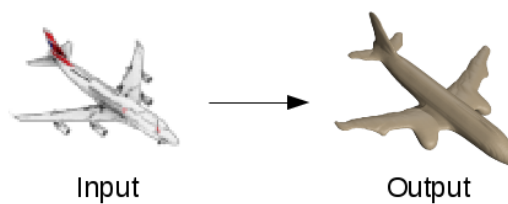


Figure 2.13: 3D Object Reconstruction from a Single Color Image [8]

### 2. Image and object classification

Image classification involves taking an image as input and determining the corresponding object class. Neural networks are particularly important in this field, as they can identify complex patterns in images and handle issues such as variations in background, angles, and poses [52].

Figure 2.14 provides an illustration of a straightforward example of image classification, featuring a dataset of images containing dogs and cats. The process involves presenting an input image of a cat to the neural network, which then utilizes its predictive capabilities to classify the image into the appropriate category [9].

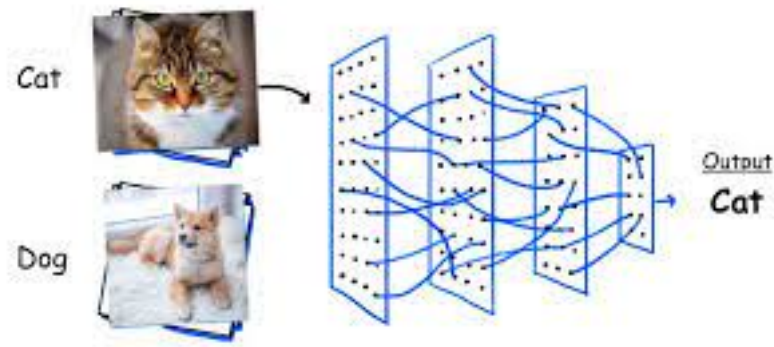


Figure 2.14: Visualization of image classification problem [9]

### 3. Object detection

This particular technique is capable of identifying objects within an input image, with detection encompassing both the ability to predict the object's class as well as its location. Localization of the object itself is achieved via regression, as the technique predicts the coordinates of the bounding box that encloses the object [40].

The figure 2.15 illustrate the bounding boxes and labels of the animals in the image [10].

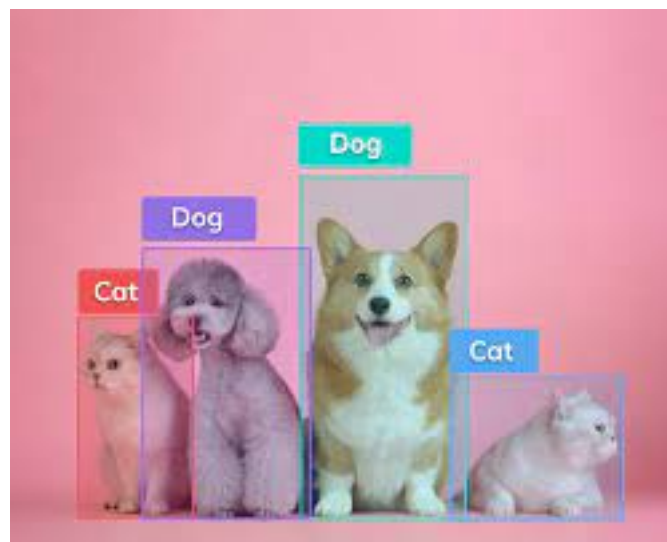


Figure 2.15: Object detection of cats and dogs [10]

### 4. Semantic segmentation

The objective of this task is to assign each pixel within an image to a specific class, providing valuable insights into the overall content of the image [40, 52].

The figure 2.16 visualize how each pixel belongs to a certain class [11].



Figure 2.16: Semantic Segmentation [11]

## 2.3 Object Detection

This thesis seeks to determine whether or not employees are adhering to necessary safety measures. Therefore, we must know both the worker's location and whether he or she is donning a helmet and a safety vest. The problem with image classification methods is that they cannot provide sufficient information when there are multiple workers in an image. Semantic segmentation provides information about the entire image, thereby increasing the training and annotation costs. Additionally, each pixel must only belong to one class. Object detection thus appears to be the optimal solution to our dilemma. Object detection algorithms are capable of classifying and localizing multiple objects within an image [54].

Object detection as a computer vision task has exist many years. Viola and Jones proposed in 2001 a detector [55] that was capable of recognizing faces, and after that, the Deformable Part-based Model proposed by Felzenszwalb et al. in 2010 [56] has the ability to extend to more complex structures. Despite the fact that these algorithms were highly successful for their period, they cannot compete the current algorithms that are based on deep Convolution neural networks [57]. These algorithms can be divided into 2 categories. Multistage object detection methods generate a set of candidate regions of interest in an image, typically in the form of bounding boxes that potentially contain objects of interest. These candidate regions are then processed by a convolutional neural network (CNN) for classification. In contrast,



single-stage detectors predict both the object category and its corresponding bounding box coordinates simultaneously, resulting in faster processing times but potentially lower accuracy compared to multistage methods [54]. The following is an examination of well-known algorithms in the age of deep convolution neural networks.

### 2.3.1 Two-stage Detectors

Two-stage detectors are another type of object detection algorithm that use a two-stage process to detect objects in an image. These detectors typically use a region proposal network (RPN) in the first stage to produce a list of potential locations for objects in the image [54]. The detector then refines these potential locations and gives each object a class probability. Two-stage detectors are known for their accuracy [58]. They work especially well in situations when precise localisation is crucial [59]. Due to the utilization of the region proposal network and the potential need for more training data and longer training times, two-stage detectors may be slower and more complex than one-stage detectors [60].

- **R-CNN [61]**

The process of RCNN begins with the region proposal generator which utilizes the selective search [62] algorithm to produce a segmentation map of the input image [54]. This map is used to generate 2000 bounding boxes that could potentially contain an object. These regions of the image, surrounded by the bounding boxes, are then fed as input to a feature extractor which is based on the CNN architecture of AlexNet [48]. The feature maps produced by the CNN are transformed into a vector using a fully connected layer, which is then classified by the support vector machine algorithm [61]. A trained Support Vector Machine is used for each class, resulting in a probability score for each class. Finally, the bounding boxes with the highest score is selected from those with high matching regions and belonging to the same class [61].

- **SPPNet [63]**

With the same accuracy as RCNN, SPPNet was able to handle images of different sizes [57]. Because in CNNs the size of the feature maps depends on the size of the input image, different image sizes require fully connected layers with the corresponding number of input neurons in order to produce the representation vector [63]. The authors of sppnet are replacing the last pooling layer with a spatial pyramid pooling

layer [64], which regardless of the feature map dimensions produces a constant size vector that will be fed into the fully connected layer [63].

- **Fast-RCNN [65]**

Fast RCNN shares the same object proposal method with RCNN, which is the selective search algorithm [62, 65]. In order to increase the speed of the model, the authors fed only the input image through the convolutional neural network, in contrast to the RCNN, which fed every object proposal [54]. The convolution and pooling layers of the CNN produce a feature map. For each region of interest (RoI), the model extracts the corresponding part of the feature map. The authors then swap out the final layer for a region of interest pooling layer to handle different image sizes [65, 63]. This layer performs max pooling with kernel width equal to the width of the RoI feature map divided by a hyperparameter  $W$ . Respectively, for the kernel height is hyperparameter  $H$  [65]. This max pooling produces a constant-size feature map with width equal to  $W$  and height equal to  $H$ . Next the fixed-size vector with length  $H \cdot W \cdot Depth_{RoIFeatureMap}$  is fed into two fully connected layers (FC) [65]. The first layer produces a softmax probability vector with length equal to the number of classes plus one for the background. The second outputs the corresponding bounding box coordinates for each class [65].

- **FPN [66]**

Prior to the introduction of the feature pyramid network for object detection, deep learning object detectors relied solely on the output of the convolutional neural network for making predictions [57]. Although this feature map contains valuable information for object classification, it poses challenges in terms of localization [57]. In response to this issue, the authors of the feature pyramid network proposed a solution that involves making predictions on different feature map scales [66]. By utilizing this approach, the model is capable of detecting objects of varying sizes and achieving better localization accuracy. As a result, the feature pyramid network has become a widely adopted backbone architecture in various object detectors [66].

### 2.3.2 One-Stage Detectors

Single-stage detectors are object detection models that predict both the object class and the related bounding box coordinates using a single convolutional neural network (CNN)

without generating any region recommendations [54]. These detectors are more computationally efficient than multistage detectors and use less memory, making them appropriate for real-time applications [60]. However, because to the inherent difficulties in precisely predicting object borders and differentiating objects from their background in a single pass, single-stage detectors often have poorer accuracy than multistage counterparts [54, 57].

- **YOLO [67]**

Yolo addresses the object detection as a regression problem [67]. The model predicts the probability of each class and the corresponding bounding boxes together, enabling it to process 45 frames per second with the graphics processing units of the time [57]. The method divides the input image into a  $S \cdot S$  grid, and each grid must make  $B$  vector predictions. These vectors have length equal to  $B \cdot 5 + C$ , where 5 is the bounding box coordinates plus the confidence score and  $C$  is the number of classes [67]. The confidence score is a value with a range from zero to one and describes how confident the model is that there is an object inside the bounding box. So the final output of YOLO is a  $S \cdot S \cdot (B \cdot 5 + C)$  tensor [67]. In the case of predictions of the same class with high matching between the bounding boxes, the model keeps the one with the highest confidence score [67]. The authors describe this method as extremely fast but with difficulties detecting small objects effecting the accuracy score [67]. New versions of YOLO [23, 68, 15] have increased the accuracy of the models while keeping the detection process in high frames per second.

- **SSD [69]**

Single shot multi box detector was able to identify objects with the same accuracy as two-stage detectors while processing more frames per second [70]. The pipeline begins with a VGG16 feature extractor that produces the feature map [69]. Then a series of convolutional layers are applied. Each layer is making predictions on the output feature map, which is divided into cells like YOLO did in the input image [70]. The high-resolution feature maps predict small objects, while the low-resolution maps predict large objects [70]. In order to increase the speed and accuracy of the model, the authors are using default bounding box sizes that are manually selected. The size of the bounding box also depends on the feature map resolution [69].

- **CenterNet [71]**

CenterNet differs from previous methods as it tries to predict the center of the object along with its width, height and category rather than relying on predefined bounding boxes [71, 57]. It utilizes CNN as the feature extractor, and the goal is to produce a heatmap for each class [71]. Each element of the maps can take two values. 1 if this point is the center of an object, or 0 if it belongs to the background. Every element of the heatmap corresponds to many pixels in the input image because the heatmaps have smaller dimensions [71]. In order to find the real center of the object in the image, the network predicts the offset variable. So by combining the center coordinates with the offset, height and width, the model predicts the bounding box of the object [71].

## 2.4 Self Supervised Learning in Computer Vision

Supervised learning for computer vision requires extensive annotated data, resulting in high costs and time investments [72]. Additionally, it may struggle with generalizing to new, untrained data [72]. A promising alternative is self-supervised learning, which uses unlabeled data and generates its own labels that will be used as the ground truth [73]. This effectively transforms unsupervised problems into supervised ones [73]. Self-supervised learning allows AI systems to learn from vast amounts of data and recognize subtle, infrequent representations but requires proper learning objectives to derive useful supervision from the data [74]. Self supervised learning can also be used as a pre-training method [73].

- **SimCLR [75]**

SimCLR is one of the best self-supervised learning methods, with higher accuracy than many supervised algorithms [76]. It uses the technique of contrastive learning [77, 76] in which the model identifies similarities between the data. The network begins with an image  $x$  pass through an encoder following by a fully connected layer that outputs a representation vector  $z$ . The encoder in the case of SimCLR is the CNN ResNet [7, 75]. For every image  $x$ , SimCLR generates a pair  $x_i, x_j$  of augmented versions. A batch with  $N$  images now becomes a new batch with  $2N$  elements [76]. The network output of  $x_i, x_j$  is  $z_i, z_j$  respectively. So the network is trained to minimize the distance between  $z_i, z_j$  and maximize the distance between them and the rest  $2(N-1)$  samples [75].

- **BYOL [78]**

BYOL aims to predict the features of one view of an image based on the features of another view of the same image [78]. This approach overcomes the limitation of negative examples used in other self-supervised learning methods, such as SimCLR [79]. The BYOL framework consists of an online network and a target network. The online network is trained to predict the features of one view of the image [78]. The target network provides stable targets for the prediction task based on another view of the same image [78]. The views of the image is obtained through data augmentations, such as random cropping and color jittering. Instead of copying the exact parameters of the online network to the target network, a moving average is taken of the online network's parameters over time, and these averaged parameters are used as the parameters for the target network [78].



# Chapter 3

## Examined Models

### 3.1 Faster-RCNN

Faster R-CNN [24] is a popular object detection algorithm that was proposed by Ren et al. in 2015. Is a better version of Fast-RCNN [65] due to the implementation of the region proposal network. This network generates object proposals based on the feature extractor output. This approach is much faster than the selective search [62, 65] algorithm used in previous versions. Also, the authors introduce anchor boxes, increasing the accuracy and speed of the model. These boxes with multiple scales and aspect ratios suggest possible objects of different sizes without scaling the input image [24].

The architect of Faster-RCNN begins with a CNN as the feature extractor [24]. In the experiments, two backbones were tested: the Zeiler and Fergus model [80] and the VGG-16 [50] CNN. Then the region proposal network takes as input every possible  $n \times n$  window of the feature map [81]. A convolution kernel with  $n \times n$  size is applied, and after that, two  $1 \times 1$ . One for the box coordinates and one for the box objectness [24]. For each spatial window, the network predicts  $k$  boxes of different scales and aspect ratios along with the objectness of the boxes. So the length of the output is  $2 \cdot k$  for objectness (2 classes object or not) and  $4 \cdot k$  for box coordinates (center(x,y) and width ,height) [81]. Every feature map pixel corresponds to a coordinate on the input image. So if the feature map height and width are  $H$  and  $W$  respectively, the network predicts in total  $H * W * k$  boxes [24].

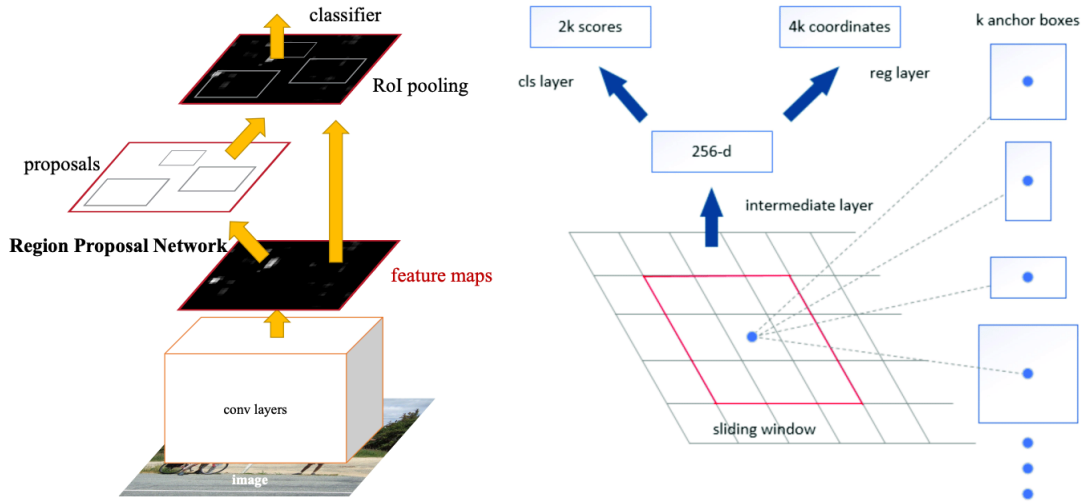


Figure 3.1: Left : Faster-RCNN network[12] , Right : Region Proposal Network [13]

The anchor boxes fall into one of two categories: object or not. If the intersection over union between an anchor and the ground truth box is higher than 0.7, the anchor is categorized as object [81]. Also, the anchors with the highest IoU with ground truth boxes are classified as positive. Negatives are the ones with IoU lower than 0.3. The rest of them are ignored by the network. In the training process, from each image, 128 positive and 128 negative samples are selected [81]. The authors suggested a ratio of 1:1 to avoid bias toward the negative samples. The 3.1 is the loss function for the Region Proposal network [24].

$$L(p_i, \mathbf{t}_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(\mathbf{t}_i, \mathbf{t}_i^*) \quad (3.1)$$

In 3.1  $p_i$  is the probability of the anchor  $i$  being an object.  $p_i^*$  is the ground truth label, 1 for positive and 0 for negative.  $L_{cls}$  is the log loss [82] and  $L_{reg} = R(\mathbf{t}_i - \mathbf{t}_i^*)$  is the regression loss, where  $R$  the smooth  $L_1$  defined in [65, 24]. The authors multiply the regression loss with  $p_i^*$  in order to activate only the positive anchors [24]. The box prediction  $\mathbf{t}_i$  consists of the vector  $[t_x, t_y, t_w, t_h]$  where  $\mathbf{t}_x = (x - x_a)/w_a$ ,  $\mathbf{t}_y = (y - y_a)/h_a$ ,  $\mathbf{t}_w = \log(w/w_a)$ ,  $\mathbf{t}_h = \log(h/h_a)$  [24].

During the training of the network the  $\mathbf{t}_i$  tries to approximate the target  $\mathbf{t}_i^*$  with elements:  $\mathbf{t}_x^* = (x^* - x_a)/w_a$ ,  $\mathbf{t}_y^* = (y^* - y_a)/h_a$ ,  $\mathbf{t}_w^* = \log(w^*/w_a)$ ,  $\mathbf{t}_h^* = \log(h^*/h_a)$  [24].  $x, y$  are the coordinates of the box center and  $w, h$  its width and height respectively. Variable  $x$  is the predicted value,  $x_a$  is for the anchor box and  $x^*$  for the target (same goes for  $y, w$ , and  $h$ ) [24]. The Faster-RCNN shares the same detection network as Fast-RCNN [65] described in 2.1.3.



## 3.2 YOLO-V7

YOLO-V7 [15] proposed by Chien-Yao Wang et al. in 2022 is a state of art real time object detector. Model scaling means make the network smaller or larger by changing some of these factors: input image size ,number of layers (depth) , number of channels (width) and the number of stages in the feature pyramid [15]. In order to make a deeper neural networks the authors suggest that is possible by controlling that shortest longest gradient path. So they proposed E-ELAN (Extended efficient layer aggregation networks) based on ELAN [83]. ELAN is illustrated in 3.2. The computational block of ELAN performs convolution between the input and a  $1 \times 1$  , two  $3 \times 3$  and four  $3 \times 3$  kernels separately. The outputs are stacked together and a convolution with a  $1 \times 1$  kernel is applied [14].

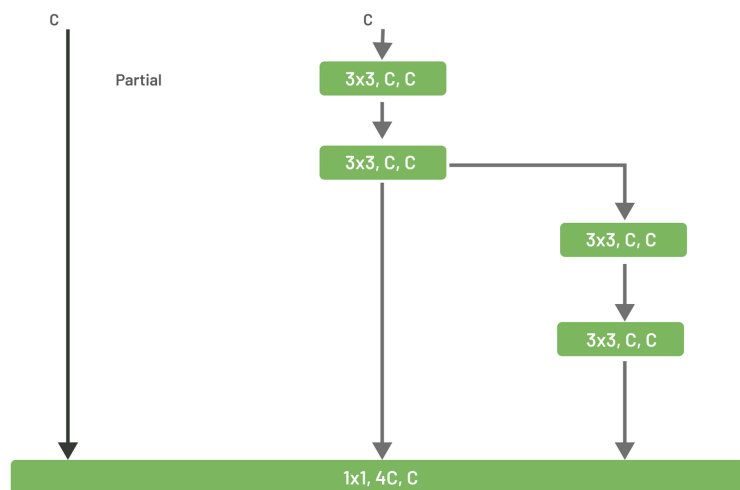


Figure 3.2: ELAN computational block [14]

E-ELAN [14] modify the computational block of the ELAN. The authors increased the number of the channels ,shuffle the outputs of different convolutions and stack them together. Then a  $1 \times 1$  convolution is applied to the concatenated feature map, performing merge cardinality [15]. The merge cardinality technique allows the network to learn a diverse set of features by using multiple paths or branches as shown in 3.3, while also keeping the computational complexity of the network manageable. By combining the feature maps from different branches, the network can capture a wider range of features and achieve better performance than a network with a single branch [15].

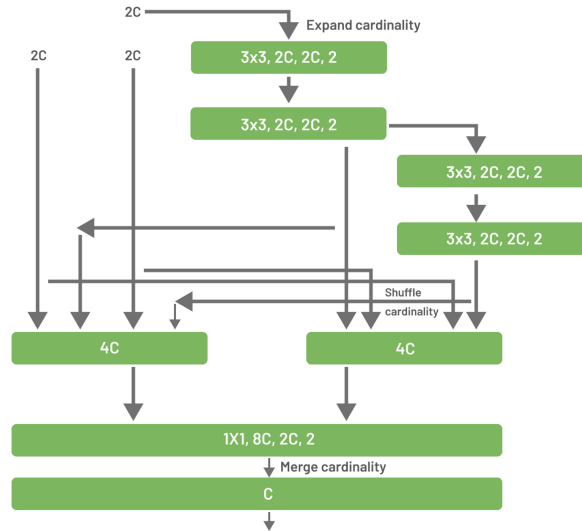


Figure 3.3: E-ELAN computational block [14]

Depending on the requirements of the object detection problem, the architecture needs to be scaled up or down [15]. Yolo-V7 is a concatenation-based model which means that feature maps from different layers are concatenated [84]. When we scale the depth in this type of models without changing the width, the efficiency of the network decreases. In order to handle this problem the authors compute how much the computing block's output channel has changed [15]. Based on this computation they scale the width of transition layers 3.4, matching the initial structure efficiency.

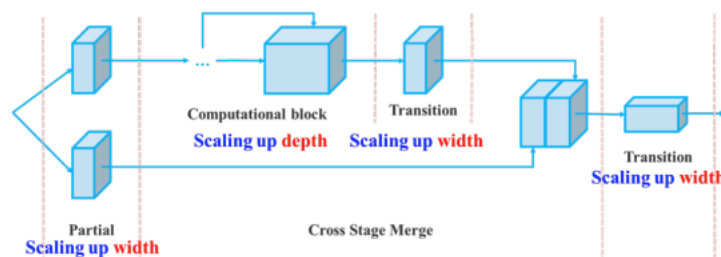


Figure 3.4: scaling width and depth on a concatenation-based model [15]

Yolo v7 utilizes two heads for prediction: the lead head, which predicts the final output, and the auxiliary head, which helps with the training of the network. Additionally, the authors implemented some tricks called "bag of freebies" to improve the training process and detection. One of these tricks is EMA [85], which is used to create the inference model by taking the average of the weights during training [15].

### 3.3 DETR

The Detection transformer was proposed by Carion et al. in 2020 [16]. DETR has a simple pipeline without the need of pre or post processing algorithms like non max suspension[86] and anchor boxes. It is based on the transformers [87], a known architect for sequence prediction. The network predicts a list of bounding boxes with the corresponding object classes simultaneously. Then bipartite matching is applied in order to assign the predictions with the annotated labels. The predicted bounding boxes with no match are labeled as no object [16].

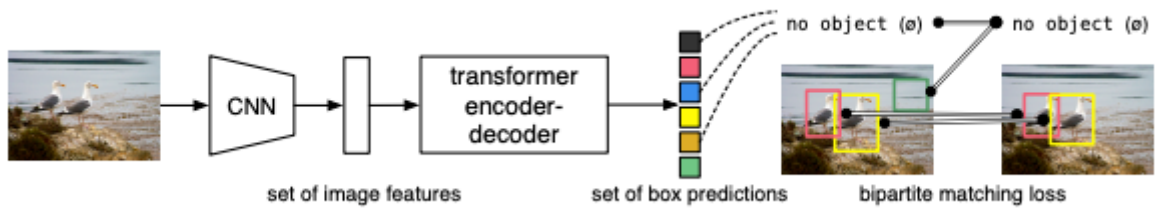


Figure 3.5: DETR pipeline [16]

Detr outputs a list of  $N$  predictions, significantly higher than the average number of annotations per image. Before the loss computation, a matching algorithm between the  $N$  predictions and ground truths is applied. The algorithm searches the order of the  $N$  elements  $\sigma \in G_N$  that minimize the cost [88, 16]:

$$\hat{\sigma} = \underset{\sigma \in G_N}{\operatorname{argmin}} \sum_i^N L_{\text{match}}(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)}) \quad (3.2)$$

Where [16]:

- $\mathbf{y}_i = (c_i, \mathbf{b}_i)$  is the ground truth box.  $c_i$  for the target class and  $\mathbf{b}_i$  for the coordinates. The authors fill the  $y$  with no object until its length becomes equal to  $N$ .
- $\mathbf{y} = \{\hat{\mathbf{y}}_i\}_{i=1}^N$  the  $N$  predictions.
- $L_{\text{match}}(\mathbf{y}_i, \hat{\mathbf{y}}_{\sigma(i)}) = -1_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \emptyset\}} L_{\text{box}}(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)})$ .  
 $\hat{p}_{\sigma(i)}(c_i)$  is the probability of the prediction with index  $\sigma(i)$  belongs to class  $c_i$  and  $\emptyset$  is the no object class.

Then the Hungarian loss [88, 16] is computed, based on the optimal order produced by the matching algorithm.

The architecture begins with the backbone ResNET50 which produces a feature map 32 times smaller than the input image and with 2048 channels (depth). The transformer encoder convolve the backbone output with a 1x1 convolution kernel, reducing the depth from 2048 to  $d$ . Then a flatten operation is performed on each channel transforming the feature map dimension from  $d \times FeatureM_{Width} \times FeatureM_{Height}$  into  $d \times FeatureM_{width} \cdot FeatureM_{Height}$  [16]. On the input of every attention layer, a fixed position encoding is added [89]. Also it is possible to stack many encoder layers.

The decoder in the Detection Transformer (DETR) uses the same architecture of the transformer [87], but decodes  $N$  objects simultaneously at each layer rather than predicting one element at a time [16]. The  $N$  input embeddings are object queries, which are learned positional encodings appended to each attention layer's input [16]. A feed forward network transforms the object queries into an output embedding, which is then decoded into box coordinates and class labels, generating  $N$  final predictions [16]. The model employs self- and encoder-decoder attention to reason globally about all objects collectively and utilize the entire image as context [16].

The final layers of detr are a feed forward network. It consisting of 3-layer perceptron with Relu as the activation function. The hidden layer with length equal to  $d$  is followed by a linear projection layer [16]. So the feed forward network outputs the class probabilities using the softmax activation function along with the 4 values that identify the bounding box (coordinates  $x,y$  of the center , width ,height) [16].

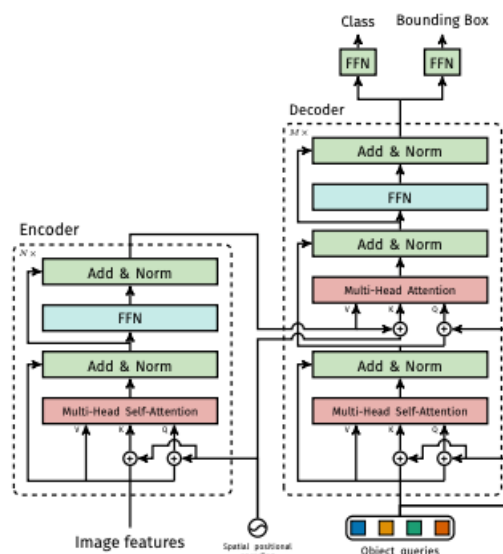


Figure 3.6: Architecture of DETR [16]

### 3.4 DINO

DINO [17] was proposed by Dosovitskiy et al. in 2021. The DINO algorithm trains without labels a vision transformer [90] or the CNN ResNET50 [7] using self-supervised learning. The training process utilizes two networks with the same architecture called student and teacher. Most self-supervised learning methods that involve knowledge distillation employ a pre-trained teacher network [91, 92, 17]. This network transfers the knowledge gained from the previous training to the student which is a smaller network. In contrast to them, DINO teacher is not pre-trained and its weights are updated from the average weights of the student network [17].

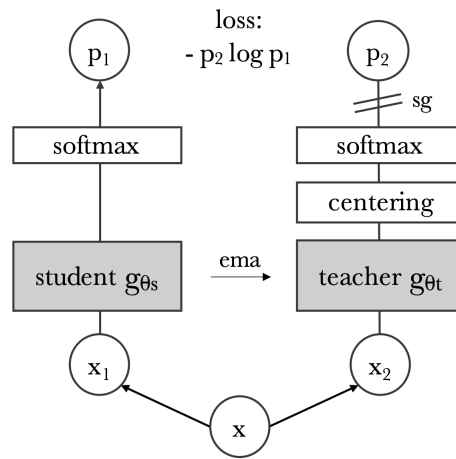


Figure 3.7: DINO student and teacher networks [17]

During the training the student tries to match the output of the teacher network. Both of them have a  $K$  dimensional probability distribution output. The formula of the student output for class  $i$  is :

$$P_s(x)^{(i)} = \frac{\exp(g_{\theta_s}(x)^{(i)}/\tau_s)}{\sum_{k=1}^K \exp(g_{\theta_s}(x)^{(k)}/\tau_s)} \quad (3.3)$$

The output is not equal to  $g_{\theta_s}(x)$  as we expected because the authors normalize the output with softmax function. In 3.3 [17]:

- $x$  is the input image. In each network, a different augmented version of  $x$  is fed.
- $g_{\theta_s}$  the student network with  $\theta_s$  parameters.
- $\tau_s$  is a temperature parameter that regulates the output distribution's sharpness.

- The teacher formula  $P_t$  is similar to  $P_s$ .

The aim of the training is to find the parameters  $\theta_s$  that minimize the cross-entropy loss 3.4 :

$$\min_{\theta_s} H(P_t(x), P_s(x)) = \min_{\theta_s} (-P_t(x) \log(P_s(x))) \quad (3.4)$$

# Chapter 4

## Experiments

### 4.1 Experiment Setup

Also all the experiments were done on google colab with the Tesla T4 and A100 graphics cards. For each experiment only one GPU was used, depending on the memory required.

### 4.2 Dataset

In recent years, ensuring human safety in the manufacturing industry has become increasingly important. With the rise of automation and advanced technologies, workers are often exposed to hazardous environments that require personal protective equipment (PPE) to minimize the risk of injury or death. Therefore, it is crucial to monitor workers and detect whether they are wearing the required PPE. To address this challenge, we find a dataset [20] that consists of images of workers in a manufacturing environment. The dataset includes five classes: helmet, no helmet, vest, no vest, and person. The purpose of this dataset is to enable the development of computer vision models that can accurately detect whether workers are wearing the required PPE. By training models on this dataset, we can help ensure the safety of workers in the manufacturing industry and reduce the risk of workplace accidents.

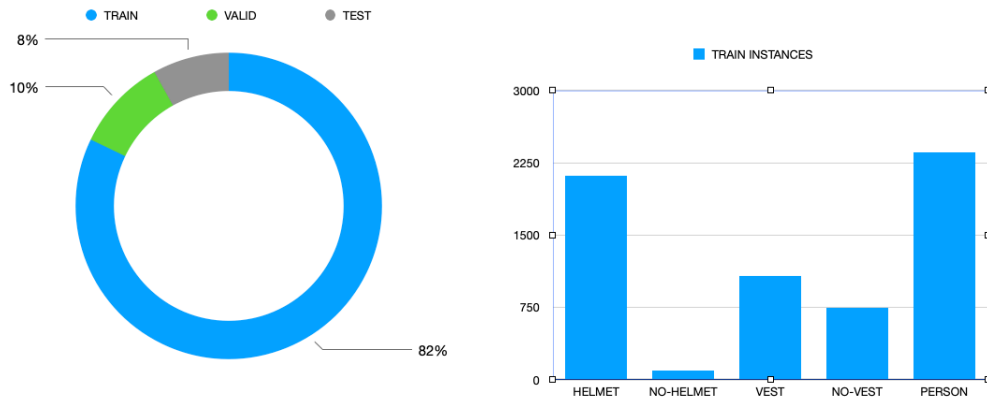


Figure 4.1: Left : train-valid-test split ,Right : Instances per class



Figure 4.2: Some examples of dataset instances

## 4.3 Settings

### 4.3.1 Metrics

Object detection algorithms are typically evaluated using the mean average precision (mAP) metric, which requires the computation of the matching area between predicted bounding boxes and their corresponding ground truth boxes. This matching area is determined through the application of the intersection over union (IoU) method, which computes the



overlap between the predicted and ground truth bounding boxes as a ratio of their union. The IoU value ranges between zero and one, with higher values indicating a greater degree of overlap and thus a stronger match between the predicted and ground truth boxes [18].

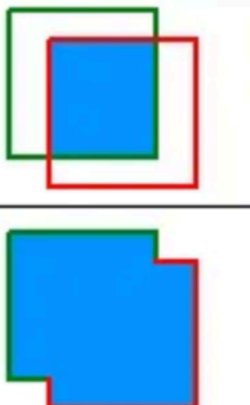
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Figure 4.3: Intersection Over Union [18]

Once we have the matching area metric, the next step in evaluating an object detection algorithm is to calculate the precision and recall [18].

$$Precision = \frac{\text{True Positive}}{\text{ALL PREDICTIONS}} \quad (4.1)$$

$$Recall = \frac{\text{True Positive}}{\text{ALL GROUND-TRUTH}} \quad (4.2)$$

Where True positives is the number of correct predictions. Every detector prediction that belongs to the same class with a ground truth box and the IoU between them is higher than a threshold, is characterized as a True Positive.

Now for each class  $i$ , we calculate the precision and recall from every image with respect to IoU threshold  $a$ . Based on the 2 columns (precision and recall), we compute the area under the precision-recall curve 4.4 referred as average precision  $AP_i@a$  [18].

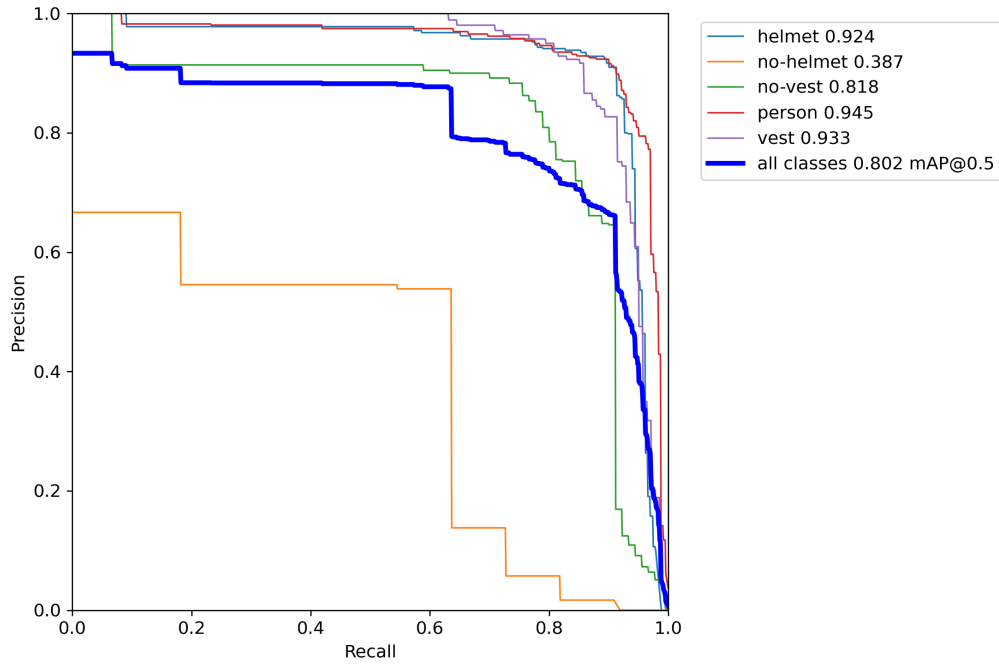


Figure 4.4: Precision-Recall curve of each class with threshold  $a = 0.5$  (from our experiments)

In order to have an overall view of the model accuracy, we compute the mean average precision over the  $n$  classes 4.3

$$mAP@_a = \frac{1}{n} \sum_{i=0}^n AP_i@_a \quad (4.3)$$

Based on the 4.3 we define the 2 metrics that we evaluate our models.

### 1. AP50

$$AP50 = mAP@0.5$$

### 2. AP

For a vector  $\mathbf{t} = [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]$  of IoU threshold values, we define as AP the following equation :

$$AP = \frac{1}{10} \sum_{j=0}^9 mAP@t_j$$

Also in the results we provide the average precision of each class for threshold = 0.5 .

## 4.3.2 Hyper-parameters

### 1. Epochs :

How many times the network will use the entire dataset for training. At each epoch, every image in the dataset is fed through the network [93].

### 2. Batch Size :

The number of input images that are fed to the network before the network update its weights [93].

### 3. Image Resolution :

The resolution at which the input image will be resized before feeding into the network. When the image resolution is empty, it means that the model automatically resizes the inputs.

### 4. Transfer Learning :

It indicates the dataset on which the network was pre-trained. When it is empty, it means that the weights are initialized based on the author's implementation.

### 5. ADAM :

Utilization of ADAM [94] optimizer.

## 4.4 Results

In this particular section, we shall present the outcomes of our experiments. For each experiment, we provide a comprehensive illustration of the corresponding losses, evaluation metrics, and the hyper-parameters employed.

### 4.4.1 Faster-RCNN Models

There are more advanced versions of Faster-RCNN than the one we analyze in 3. Our experiments use them in order to reach the highest mAP. Basically these versions have the same architecture with the original model [24] but with different backbones (ResNets CNNs with Feature pyramid network). In the below graphs the followings stats are represented:

- Loss RPN cls : Classification loss of the region proposal network.

- Loss RPN loc : Localization loss of the region proposal network.
- Loss cls : Classification loss of the Faster-RCNN head.
- Loss Box Reg : Localization loss of the Faster-RCNN head.
- Total Loss : The sum of losses.

## 1. Faster-RCNN R50-FPN

Version and Hyper-parameters :

- Version : R50-FPN
- Batch size : 4
- Transfer Learning : COCO
- Image Resolution : -
- ADAM : -

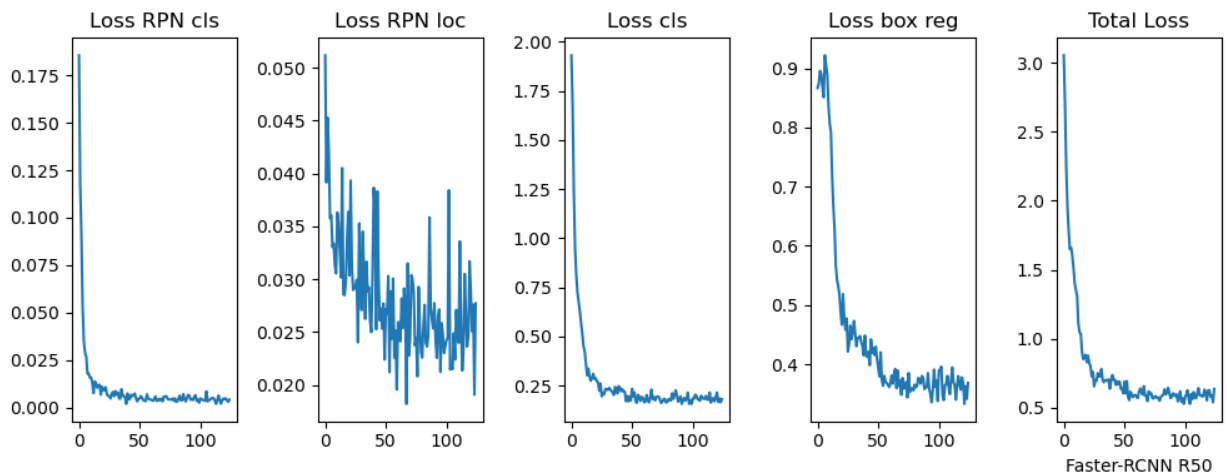


Figure 4.5: Faster-RCNN R50-FPN training stats

| MODEL        | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|--------------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| Faster R-CNN | R-50    | 41.5 | 81.4 | 60.2      | 48.6      | 23.8         | 44.6    | 30.3       | 4          | COCO              | -         |

Table 4.1: Faster-RCNN R50 FPN results and hyper-parameters

## 2. Faster-RCNN R101-FPN

Version and Hyper-parameters :

- Version : R101-FPN
- Batch size : 4
- Transfer Learning : COCO
- Image Resolution : -
- ADAM : -

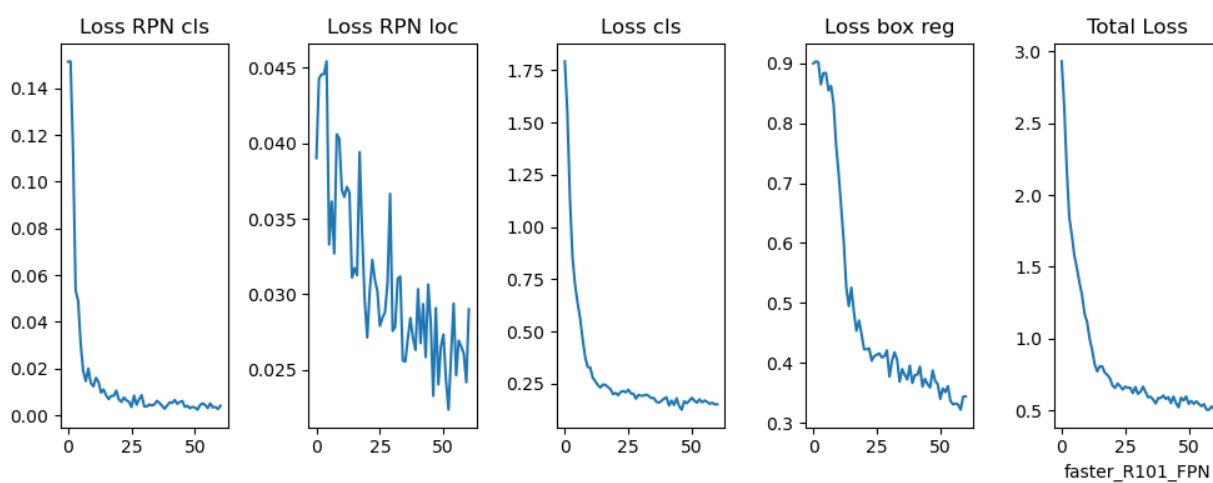


Figure 4.6: Faster-RCNN R101-FPN training stats

| MODEL        | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|--------------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| Faster R-CNN | R-101   | 35.5 | 70.0 | 58.0      | 47.8      | 6.8          | 42.6    | 22.5       | 4          | COCO              | -         |

Table 4.2: Faster-RCNN R101 FPN results and hyper-parameters

### 3. Faster-RCNN X101-FPN

Version and Hyper-parameters :

- Version : X101-FPN
- Batch size : 4
- Transfer Learning : COCO
- Image Resolution : -
- ADAM : -

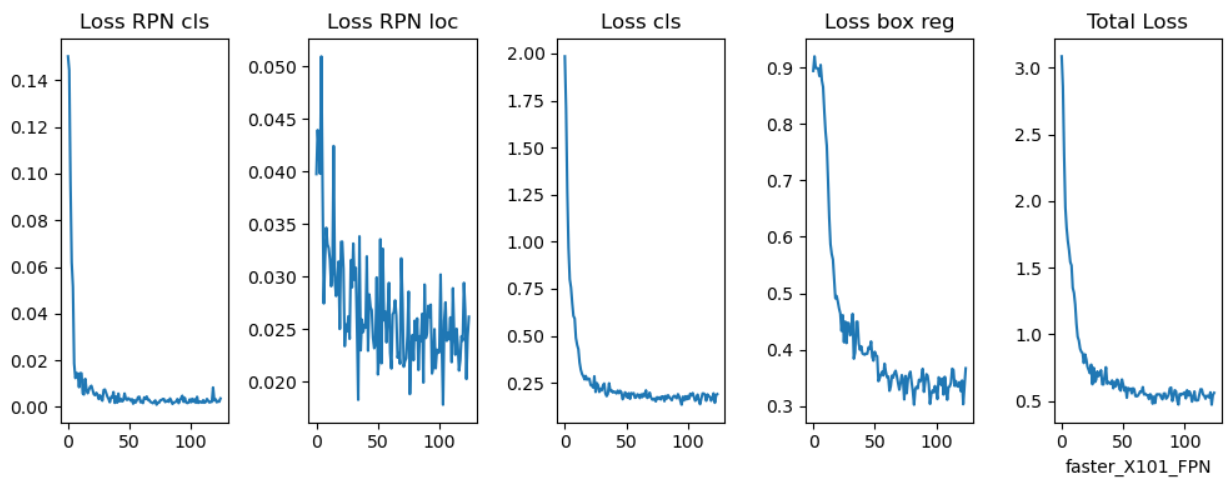


Figure 4.7: Faster-RCNN X101-FPN training stats

| MODEL        | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|--------------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| Faster R-CNN | X-101   | 39.9 | 77.7 | 59.2      | 50.0      | 16.1         | 45.4    | 28.5       | 4          | COCO              | -         |

Table 4.3: Faster-RCNN X101 FPN results and hyper-parameters

## 4.4.2 YOLO-V7 Models

In the following results, the 10 plots are : The localization loss for the train and valid dataset. The objectness loss ( how confident the model is that there is object inside the bounding box) for the train and valid dataset. The classification loss for train and valid dataset. Also the evaluation metrics (precision, recall, AP50, AP) are computed on the validation dataset.

### 1. YOLO-V7

Version and Hyper-parameters :

- Version : Tiny
- Batch size : 16
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : -

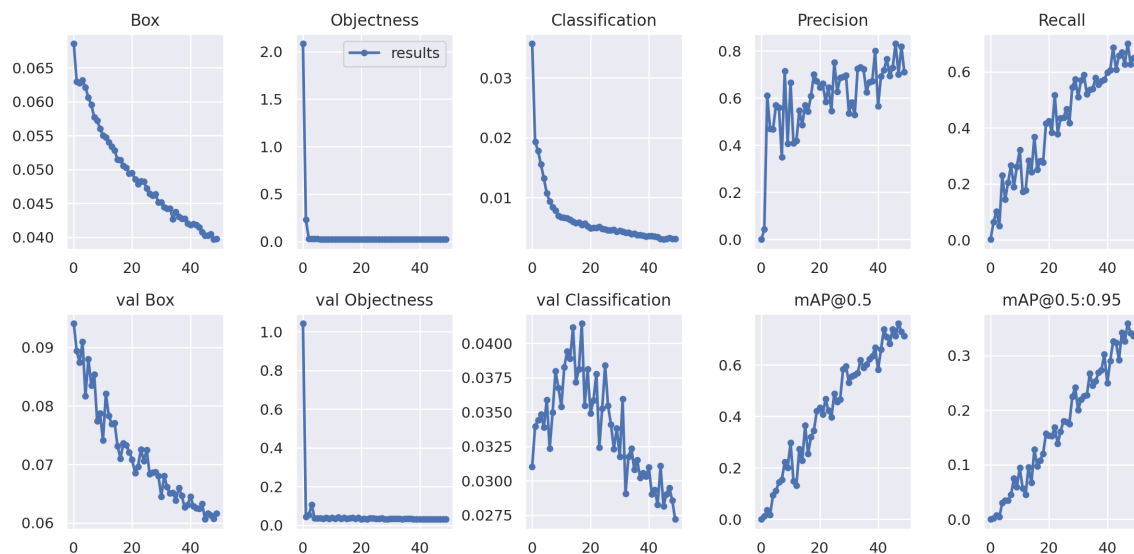


Figure 4.8: YOLO-V7 1 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | Tiny    | 28.2 | 65.4 | 40.8      | 39.3      | 7.4          | 34.6    | 19.1       | 16         | -                 | 640x640   |

Table 4.4: YOLO-v7 tiny 1

## 2. YOLO-V7

Version and Hyper-parameters :

- Version : Tiny
- Batch size : 32
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : -

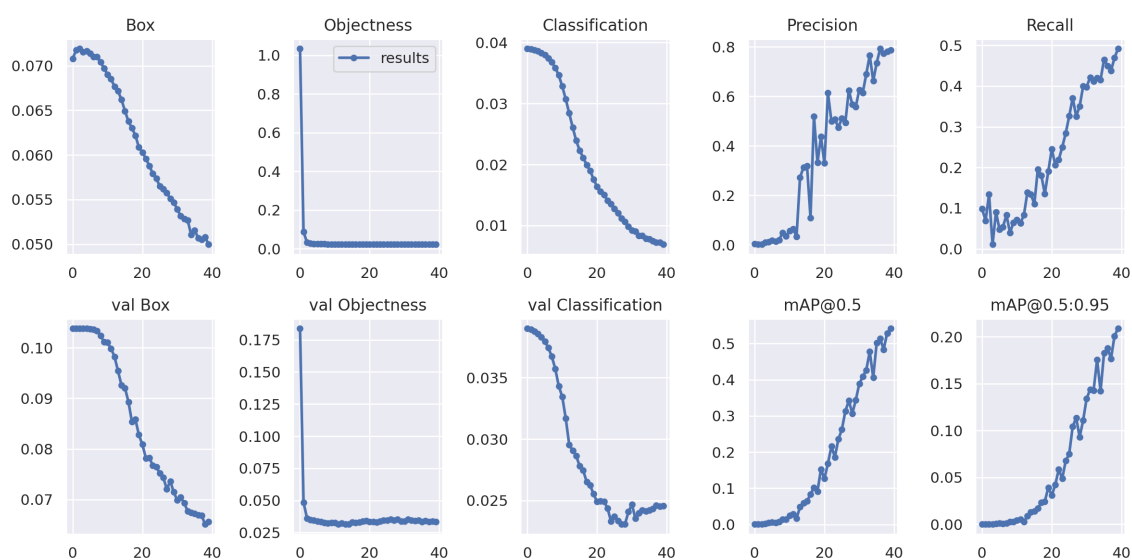


Figure 4.9: YOLO-V7 2 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | Tiny    | 18.0 | 45.3 | 27.6      | 29.5      | 0            | 27.4    | 5.48       | 32         | -                 | 640x640   |

Table 4.5: YOLO-v7 tiny 2



### 3. YOLO-V7

Version and Hyper-parameters :

- Version : V7
- Batch size : 16
- Transfer Learning : COCO
- Image Resolution : 416x416
- ADAM : -

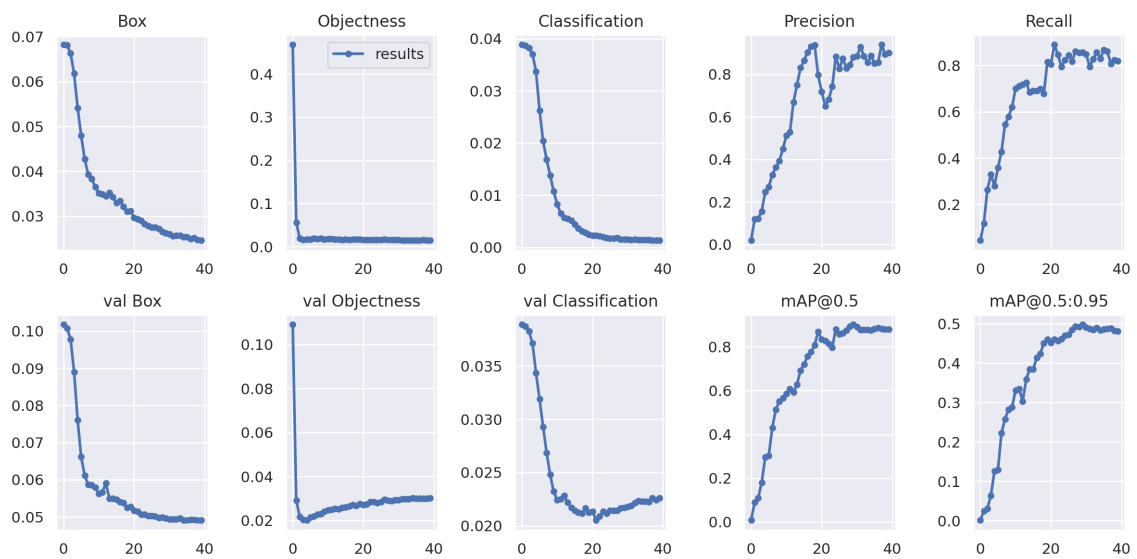


Figure 4.10: YOLO-V 73 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | v7      | 41.4 | 78.6 | 60.0      | 49.3      | 23.7         | 46.5    | 27.7       | 16         | COCO              | 416x416   |

Table 4.6: YOLO-v7 3

## 4. YOLO-V7

Version and Hyper-parameters :

- Version : V7
- Batch size : 16
- Transfer Learning : COCO
- Image Resolution : 640x640
- ADAM : -

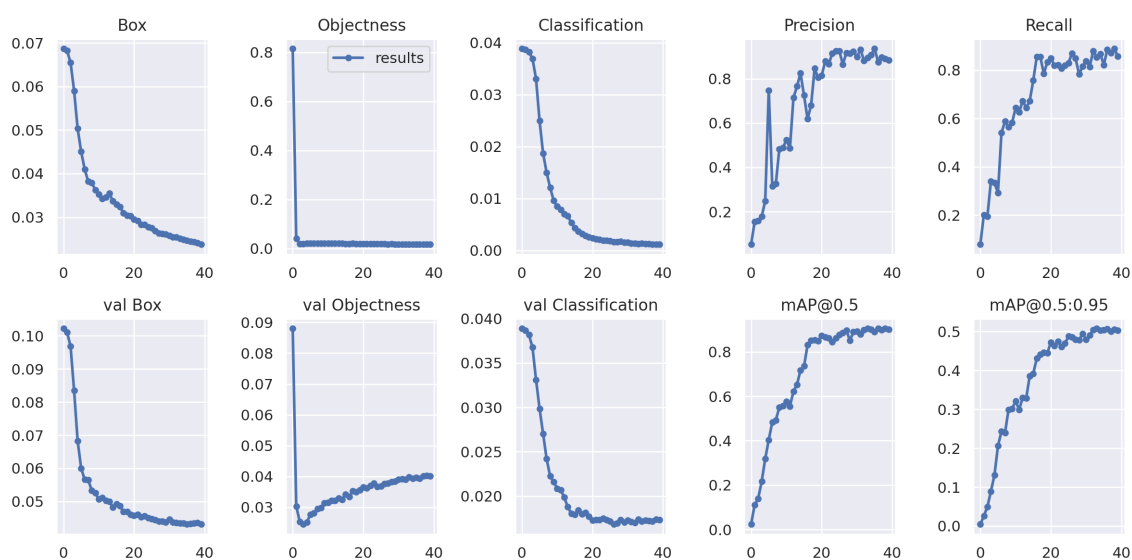


Figure 4.11: YOLO-V7 4 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | v7      | 40.7 | 77.4 | 60.1      | 46.4      | 22.7         | 46.4    | 27.7       | 16         | COCO              | 640x640   |

Table 4.7: YOLO-v7 4

## 5. YOLO-V7

Version and Hyper-parameters :

- Version : V7
- Batch size : 16
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : -

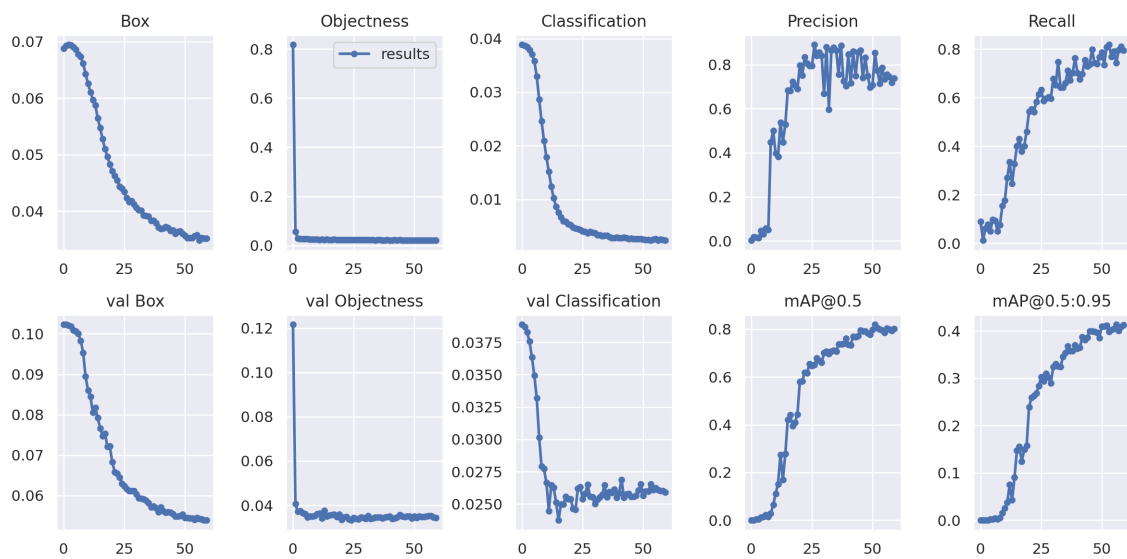


Figure 4.12: YOLO-V7 5 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | v7      | 35.4 | 70.6 | 48.9      | 44.3      | 18.5         | 41.1    | 24.0       | 16         | -                 | 640x640   |

Table 4.8: YOLO-v7 5

## 6. YOLO-V7

Version and Hyper-parameters :

- Version : V7
- Batch size : 16
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : YES
- FREEZING : YES(backbone weights are not updated)

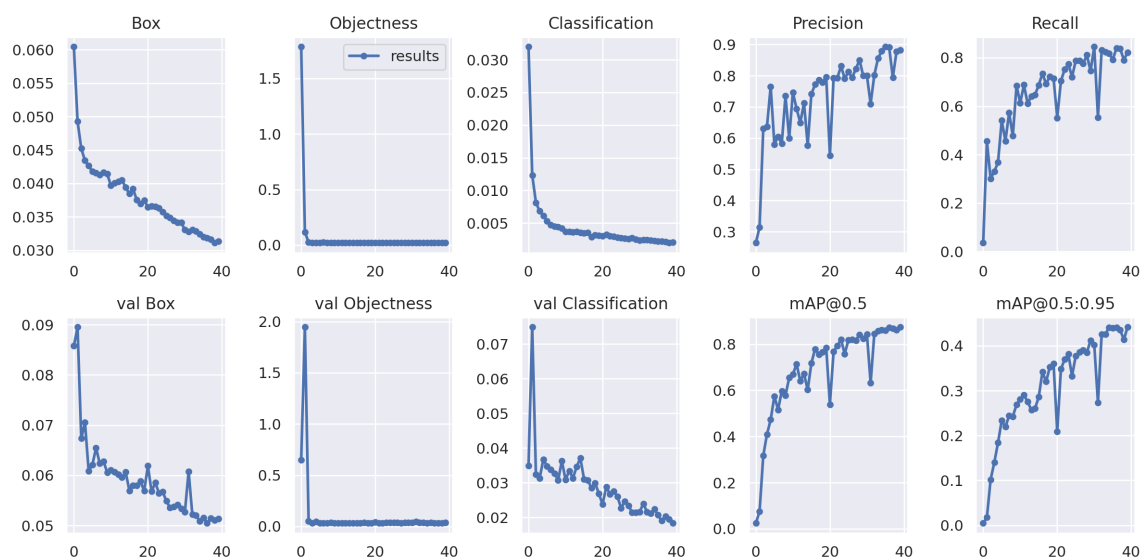


Figure 4.13: YOLO-V7 6 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | v7-ADAM | 35.3 | 73.4 | 51.3      | 47.1      | 20.2         | 39.5    | 18.3       | 16         | COCO              | 640x640   |

Table 4.9: YOLO-v7 6

## 7. YOLO-V7

Version and Hyper-parameters :

- Version : V7-X
- Batch size : 16
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : -

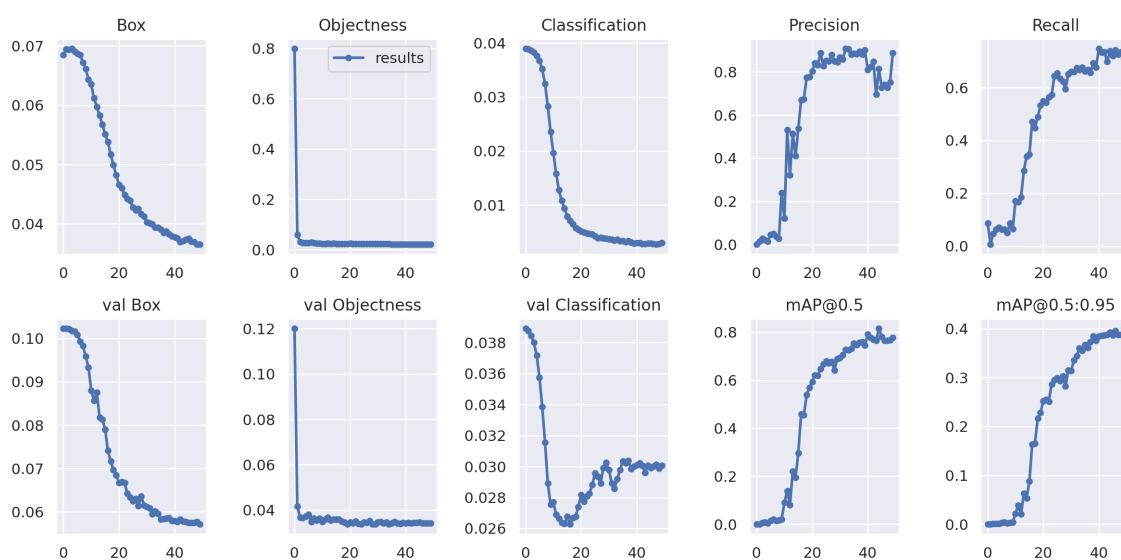


Figure 4.14: YOLO-V7X 7 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | v7-X    | 33.3 | 67.3 | 47.3      | 42.6      | 11.2         | 40.2    | 25.0       | 16         | -                 | 640x640   |

Table 4.10: YOLO-V7X 7

## 8. YOLO-V7

Version and Hyper-parameters :

- Version : V7-X
- Batch size : 16
- Transfer Learning : COCO
- Image Resolution : 640x640
- ADAM : -

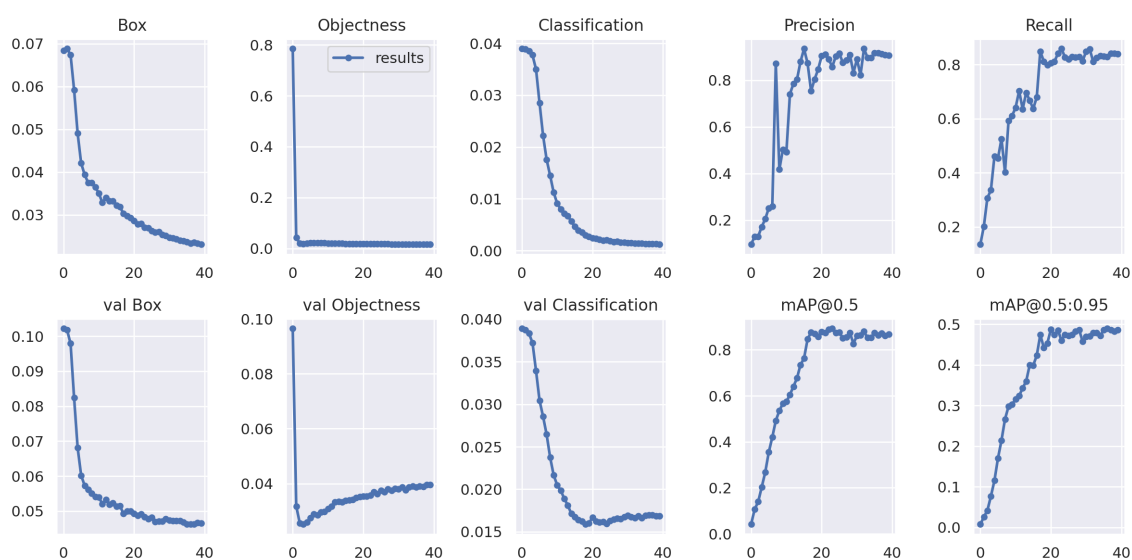


Figure 4.15: YOLO-V7X 8 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-X    | 40.3 | 75.9 | 59.7      | 48.6      | 16.0         | 43.8    | 33.2       | 16         | COCO              | 640x640   |

Table 4.11: YOLO-v7X 8

## 9. YOLO-V7

Version and Hyper-parameters :

- Version : V7-X
- Batch size : 32
- Transfer Learning : -
- Image Resolution : 640x640
- ADAM : -

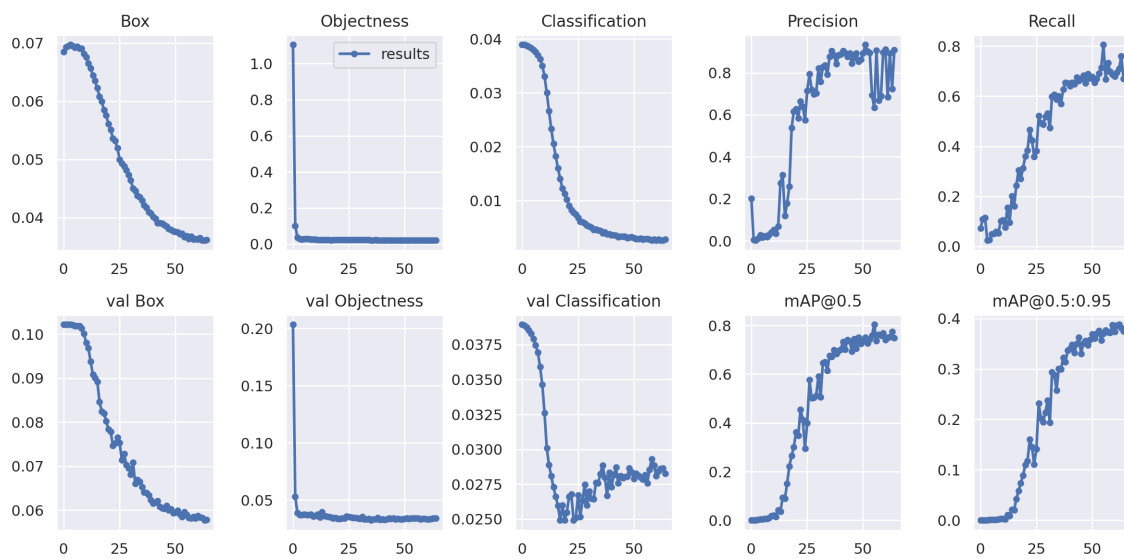


Figure 4.16: YOLO-V7X 9 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-X    | 33.9 | 69.1 | 47.1      | 41.2      | 17.4         | 38.9    | 24.7       | 32         | -                 | 640x640   |

Table 4.12: YOLO-v7X 9

## 10. YOLO-V7

Version and Hyper-parameters :

- Version : V7-X
- Batch size : 32
- Transfer Learning : COCO
- Image Resolution : 640x640
- ADAM : -

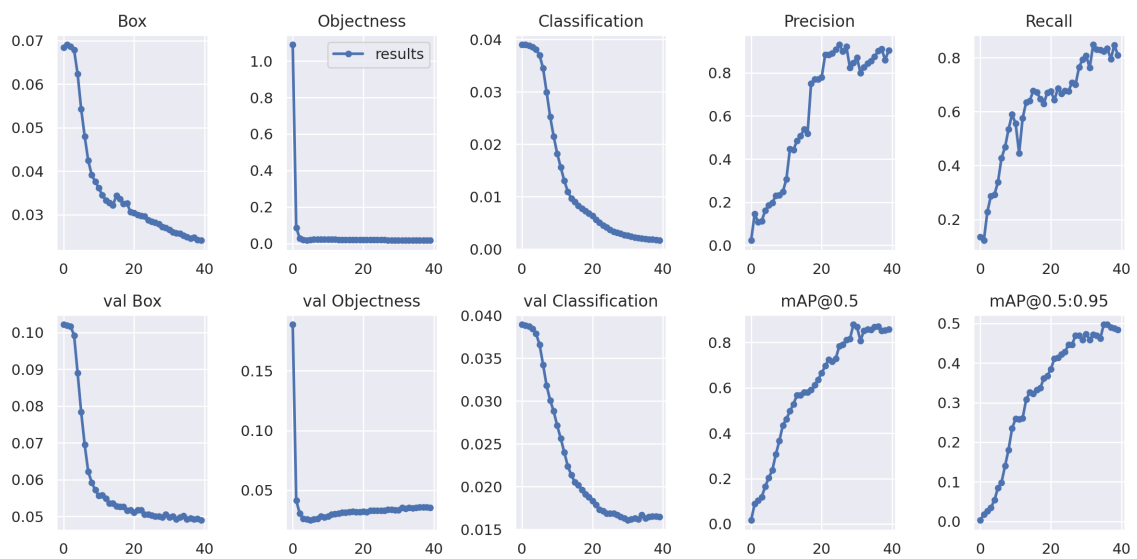


Figure 4.17: YOLO-V7X 10 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-X    | 40.8 | 78.4 | 57.6      | 50.2      | 21.0         | 41.5    | 33.6       | 32         | COCO              | 640x640   |

Table 4.13: YOLO-v7X 10



## 11. YOLO-V7

Version and Hyper-parameters :

- Version : V7-X
- Batch size : 32
- Transfer Learning : COCO
- Image Resolution : 640x640
- ADAM : YES

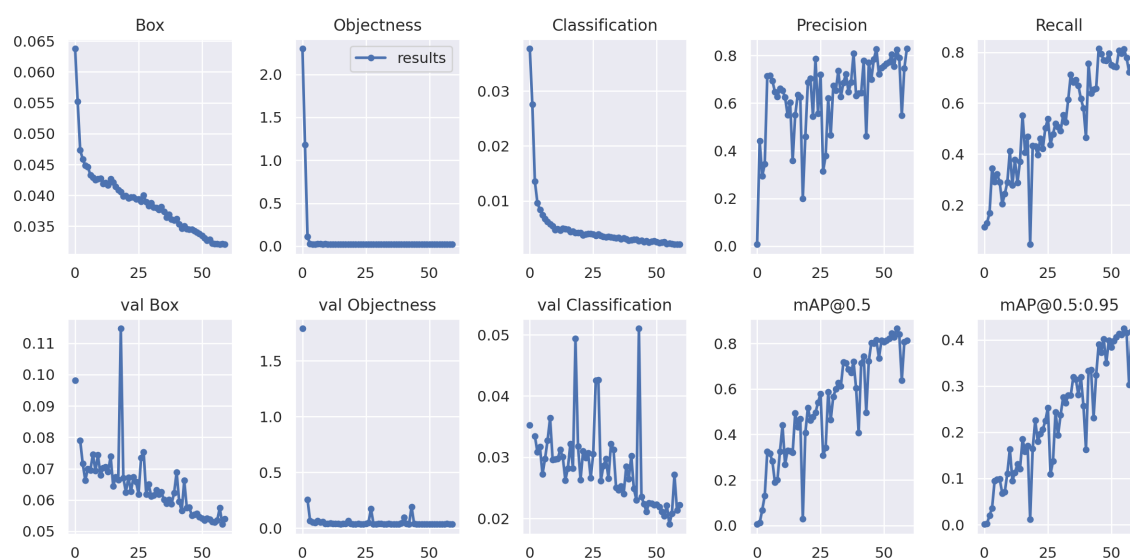


Figure 4.18: YOLO-V7X 11 training stats

| MODEL   | VERSION   | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|-----------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-X ADAM | 31.6 | 69.9 | 46.0      | 40.0      | 13.3         | 37.6    | 20.8       | 32         | COCO              | 640x640   |

Table 4.14: YOLO-v7X 11

## 12. YOLO-V7

Version and Hyper-parameters :

- Version : V7-W
- Batch size : 8
- Transfer Learning : COCO
- Image Resolution : 1280x1280
- ADAM : -

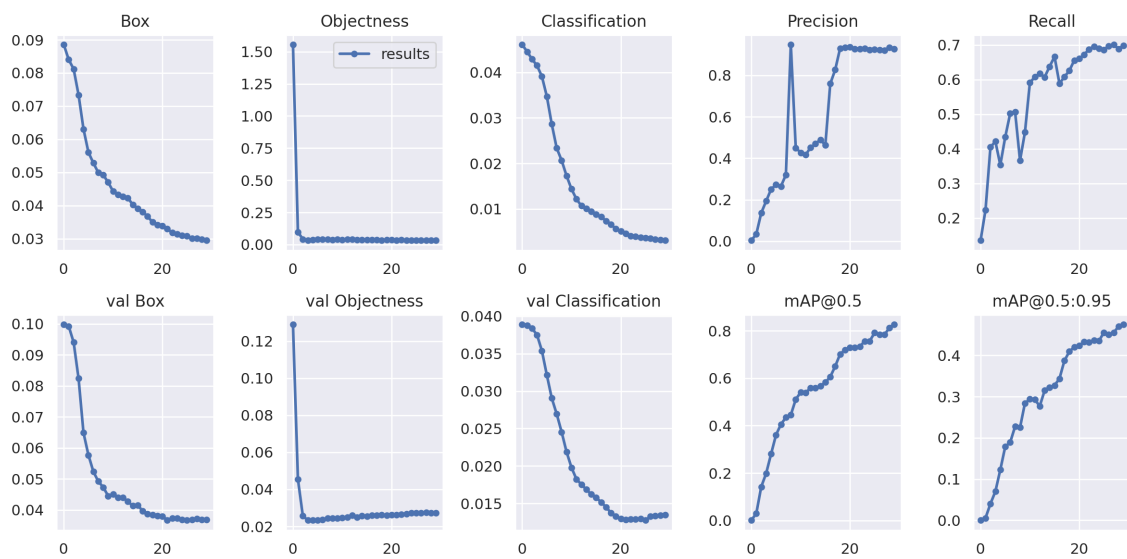


Figure 4.19: YOLO-V7W 12 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-W    | 39.8 | 74.0 | 62.1      | 47.3      | 14.8         | 45.0    | 29.7       | 8          | COCO              | 1280x1280 |

Table 4.15: YOLO-V7W 12

### 13. YOLO-V7

Version and Hyper-parameters :

- Version : V7-W
- Batch size : 16
- Transfer Learning : COCO
- Image Resolution : 640x640
- ADAM : -

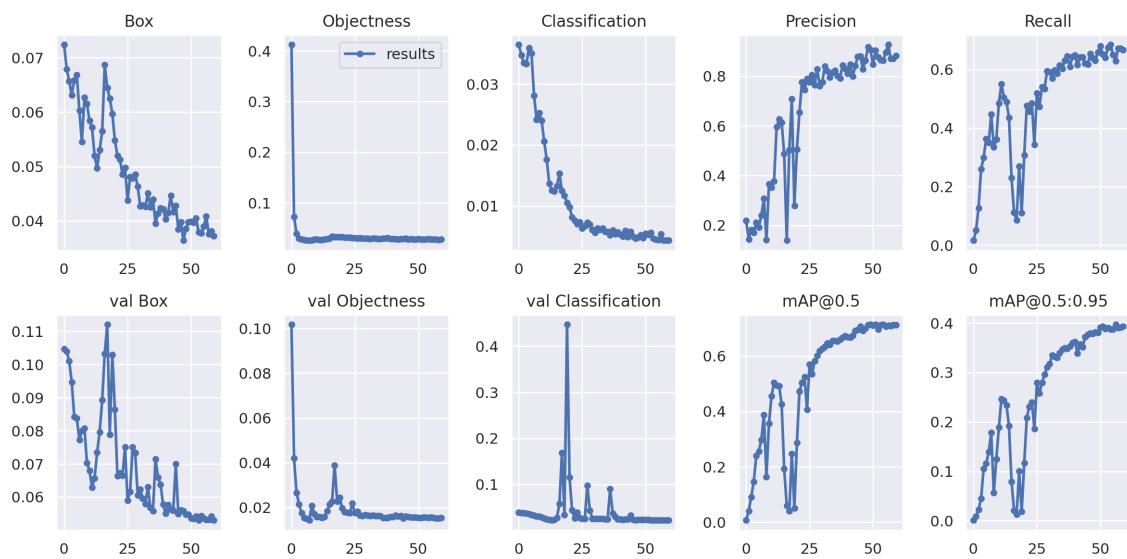


Figure 4.20: YOLO-V7W 13 training stats

| MODEL   | VERSION | AP   | AP50 | AP PERSON | AP HELMET | AP NO-HELMET | AP VEST | AP NO-VEST | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|---------|---------|------|------|-----------|-----------|--------------|---------|------------|------------|-------------------|-----------|
| YOLO-v7 | V7-W    | 32.8 | 62   | 52.8      | 45.4      | 0.3          | 41.6    | 24.0       | 16         | COCO              | 640x640   |

Table 4.16: YOLO-V7W 13

### 4.4.3 DETR

#### 1. DETR without pre-train

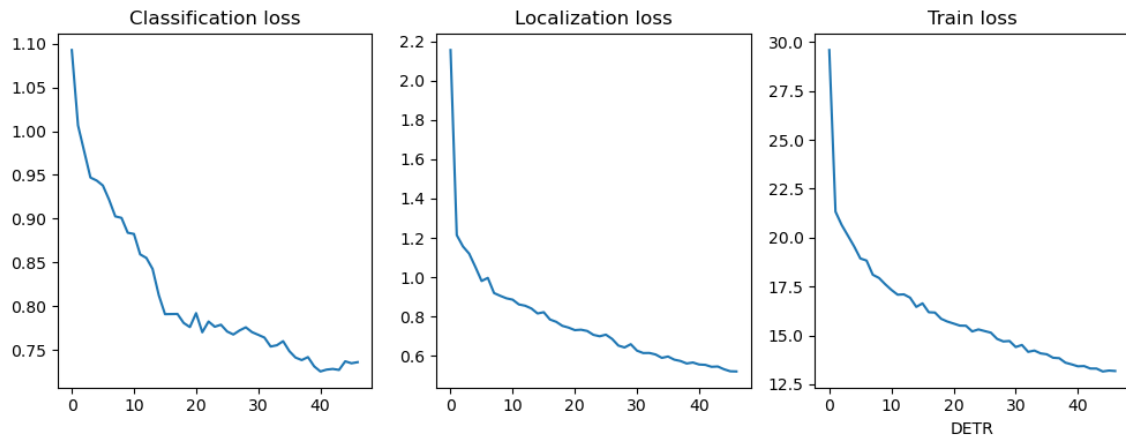


Figure 4.21: Training stats of DETR without pre-train

| MODEL | BACKBONE | AP   | AP50 | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|-------|----------|------|------|------------|-------------------|-----------|
| DETR  | ResNET50 | 3.66 | 10.8 | 4          | -                 | -         |

Table 4.17: DETR without pre-train

#### 2. Pre-trained DETR on COCO

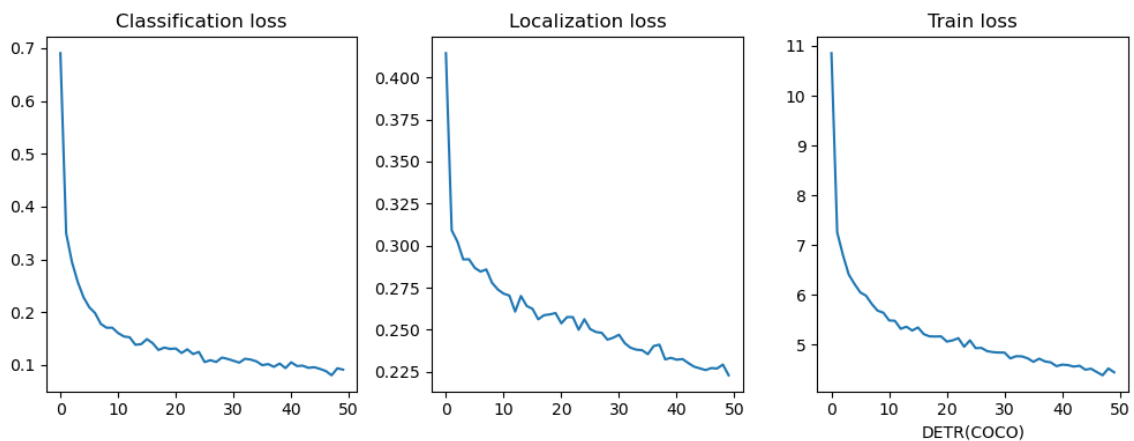


Figure 4.22: Training stats of DETR (COCO)

| MODEL | BACKBONE | AP   | AP50 | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|-------|----------|------|------|------------|-------------------|-----------|
| DETR  | ResNET50 | 47.3 | 81.5 | 4          | COCO              | -         |

Table 4.18: Pre-train DETR on COCO

### 3. Pre-trained DETR with DINO

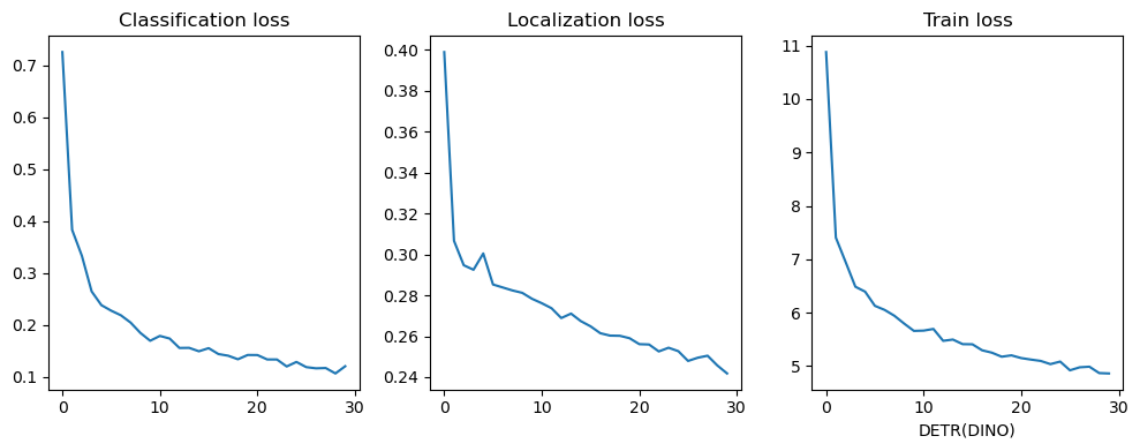


Figure 4.23: Training stats of DETR(DINO)

In this case, we use a backbone ResNET50 which is pre-trained with the self supervised algorithm DINO.

| MODEL | BACKBONE | AP   | AP50 | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|-------|----------|------|------|------------|-------------------|-----------|
| DETR  | ResNET50 | 44.2 | 78.6 | 4          | DINO              | -         |

Table 4.19: Pre-trained DETR with DINO

## 4.5 Summary

| MODEL        | VERSION   | AP            | AP50         | BATCH SIZE | TRANSFER LEARNING | IMAGE RES |
|--------------|-----------|---------------|--------------|------------|-------------------|-----------|
| Faster R-CNN | R-50      | 41.5 (-5.8%)  | 81.4(-0.1%)  | 4          | COCO              | -         |
| Faster R-CNN | R-101     | 35.5(-11.8%)  | 70.0(-11.5%) | 4          | COCO              | -         |
| Faster R-CNN | X-101     | 39.9(-7.4%)   | 77.7(-3.8%)  | 4          | COCO              | -         |
| YOLO-v7      | Tiny      | 28.2(-19.1%)  | 65.4(-16.1%) | 16         | -                 | 640x640   |
| YOLO-v7      | Tiny      | 18.0(-29.3%)  | 45.3(-36.2%) | 32         | -                 | 640x640   |
| YOLO-v7      | v7        | 41.4(-5.9%)   | 78.6(-2.9%)  | 16         | COCO              | 416x416   |
| YOLO-v7      | v7        | 40.7(-6.6%)   | 77.4(-4.1%)  | 16         | COCO              | 640x640   |
| YOLO-v7      | v7        | 35.4(-11.9%)  | 70.6(-10.9%) | 16         | -                 | 640x640   |
| YOLO-v7      | v7-ADAM   | 35.3(-12.0%)  | 73.4(-8.1%)  | 16         | COCO              | 640x640   |
| YOLO-v7      | v7-X      | 33.3(-14.0%)  | 67.3(-14.2%) | 16         | -                 | 640x640   |
| YOLO-v7      | V7-X      | 40.3(-7.0%)   | 75.9(-5.6%)  | 16         | COCO              | 640x640   |
| YOLO-v7      | V7-X      | 33.9(-13.4%)  | 69.1(-12.4%) | 32         | -                 | 640x640   |
| YOLO-v7      | V7-X      | 40.8(-6.5%)   | 78.4(-3.1%)  | 32         | COCO              | 640x640   |
| YOLO-v7      | V7-X ADAM | 31.6(-15.7%)  | 69.9(-11.6%) | 32         | COCO              | 640x640   |
| YOLO-v7      | V7-W      | 39.8(-7.5%)   | 74.0(-7.5%)  | 8          | COCO              | 1280x1280 |
| YOLO-v7      | V7-W      | 32.8(-14.5%)  | 62(-19.5%)   | 16         | COCO              | 640x640   |
| DETR         | ResNET50  | 3.66 (-43.6%) | 10.8(-70.7%) | 4          | -                 | -         |
| DETR         | ResNET50  | <b>47.3</b>   | <b>81.5</b>  | 4          | COCO              | -         |
| DETR         | ResNET50  | 44.2(-3.1%)   | 78.6(-2.9%)  | 4          | DINO              | -         |

Table 4.20: Summary of all experiments

The best model on both AP and AP50 metrics is the DETR (Pre-trained on COCO). We will try to maximize the the AP AND AP50 by searching the optimal hyper-parameters of DETR (COCO).

| AP               | AP50               | BATCH SIZE | Learning Rate(lr) | Backbone lr | Encoder Layers | Decoder Layers |
|------------------|--------------------|------------|-------------------|-------------|----------------|----------------|
| <b>47.3</b>      | 81.5               | 4          | 1e-4              | 1e-5        | 6              | 6              |
| 0.01(-47.3%)     | 0.19 (-81.3%)      | 4          | 1e-3              | 1e-5        | 6              | 6              |
| 44.9(-2.4%)      | 80.6(-0.9%)        | 8          | 1e-4              | 1e-5        | 6              | 6              |
| <b>47.(0.0%)</b> | <b>84.3(+2.8%)</b> | 8          | 5e-5              | 5e-5        | 6              | 6              |
| 41.4(-5.9%)      | 78(-3.5%)          | 4          | 1e-4              | 1e-4        | 3              | 3              |
| 2.19(-45.1%)     | 6.67 (-74.8%)      | 4          | 1e-4              | 1e-5        | 9              | 9              |
| 0.01(-47.3%)     | 0.06 (-81.4%)      | 8          | 1e-3              | 1e-4        | 9              | 9              |
| 45.4(-1.9%)      | 79(-2.5%)          | 4          | 5e-4              | 5e-6        | 6              | 6              |

Table 4.21: DETR (COCO) hyper-parameter tuning

The best model for our problem, based on the AP50, is the DETR (COCO) with 8 batch size , 5e-5 lr, 5e-5 backbone lr, 6 encoder and decoder layers. We choose the metric AP50

---

in relation to AP because we are mainly interested in the correct detection, regardless of the IoU (as long as it is greater than 0.5). First, from the results, we observe the importance of transfer learning both in increasing the mAP and in faster training of the neural network. Furthermore, it is obvious that transformers without pre-training need a large amount of data to operate. This is why, in many of our experiments, the AP and AP50 of DETR (without pre-train) are almost zero. But the pre-training does not necessarily have to be supervised. The DETR model in which we change its backbone to the DINO ResNet50 has a difference of only 2.9% (AP50) compared to the supervised on coco ResNet50. Finally, the AP of each class is highly correlated with the annotations per class.





# Chapter 5

## Conclusions

In conclusion, the manufacturing industry is facing a serious safety problem, with many deaths and injuries occurring each year. To address this issue, we have explored the potential of computer vision methods, specifically object detection, to increase employee safety. Through our experiments, we have trained various models, including YOLO V7 and Faster RCNN, and compared them to detection transformers. Our findings suggest that pre-trained DETR on COCO achieves the highest mean average precision with an IoU threshold of 0.5. Transfer learning is crucial in transformers when the training dataset is small. However, we also observed that the imbalance of annotations per class can cause low mean average precision for some classes. Overall, our study demonstrates the potential of detection transformers, in improving safety in the manufacturing industry. Our findings highlight the importance of appropriate model selection and transfer learning in achieving high precision and recall in object detection. We hope that our work will contribute to the ongoing efforts to create safer working environments in the manufacturing industry.

### 5.1 Future work

Our study has opened up several avenues for future research to build on our findings. Firstly, we suggest the creation of a new dataset with a larger number of examples and a balance between the annotations per class. This will help improve the overall performance of detection transformers and increase their accuracy in detecting objects. Secondly, we recommend exploring changes in the DETR architecture to improve the inference speed of the model. This will allow the model to process more frames per second, making it more suitable

for real-time applications in the manufacturing industry. Finally, we propose pre-training the model on a more specific dataset using the method DINO, which does not require annotations. This will enable the model to learn more abstract and general features that can be useful for detecting objects in new environments. By pre-training the model on a broader range of data, we can improve its performance and enhance its ability to detect objects in various settings. In conclusion, our study has shed light on several potential avenues for future research in the area of object detection for safety in the manufacturing industry. By continuing to explore these areas, we can further improve the performance of detection transformers and create safer working environments for employees.

# Bibliography

- [1] Understanding the machine learning workflow. <https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781783980284/2/ch021v11sec09/understanding-the-machine-learning-workflow>.
- [2] DeepAI. Perceptron. <https://deepai.org/machine-learning-glossary-and-terms/perceptron>.
- [3] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017.
- [4] V7. Activation functions in neural networks. <https://www.v7labs.com/blog/neural-networks-activation-functions#h1>.
- [5] Wolfram World. HyperbolicTangent. <https://mathworld.wolfram.com/HyperbolicTangent.html>.
- [6] IndoML. Student notes: Convolutional neural networks (cnn) introduction. <https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/>.
- [7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [8] High quality 3d object reconstruction from a single color image. <https://bair.berkeley.edu/blog/2017/08/23/high-quality-3d-object-reconstruction/>. access date: 07-04-2023.

- [9] Camp K12. Image classification using mobilenet. <https://campk12.com/project/syed.ghaleb/image-classification-using-mobile-net>.
- [10] V7 lab. The ultimate guide to object detection. <https://www.v7labs.com/blog/object-detection-guide>.
- [11] The Gradient. Going beyond the bounding box with semantic segmentation. <https://thegradients.pub/semantic-segmentation/>.
- [12] Faster r-cnn. <https://paperswithcode.com/method/faster-r-cnn>.
- [13] Denan Xia, Peng Chen (□□), Bing Wang, Jun Zhang, and Chengjun Xie. Insect detection and classification based on an improved convolutional neural network. *Sensors*, 18:4169, 11 2018.
- [14] Yolov7 architecture explanation. <https://www.cameralyze.co/blog/yolov7-architecture-explanation>.
- [15] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022.
- [16] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020.
- [17] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.
- [18] Object detection metrics with worked example. <https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e>.

- [19] Watch for falling objects: Ppe to protect you on the jobsite. <https://ohsonline.com/Articles/2021/06/01/Watch-for-Falling-Objects-PPE-to-Protect-You-on-the-Job-Site.aspx?Page=2>.
- [20] construction safety image dataset. <https://universe.roboflow.com/roboflow-100/construction-safety-gsnvb/dataset/2>.
- [21] Fangbo Zhou, Huailin Zhao, and Zhen Nie. Safety helmet detection based on yolov5. In *2021 IEEE International conference on power electronics, computer applications (ICPECA)*, pages 6–11. IEEE, 2021.
- [22] Fan Wu, Guoqing Jin, Mingyu Gao, HE Zhiwei, and Yuxiang Yang. Helmet detection based on improved yolo v3 deep model. In *2019 IEEE 16th International conference on networking, sensing and control (ICNSC)*, pages 363–368. IEEE, 2019.
- [23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [25] Ran Shao and Xiao-Jun Bi. Transformers meet small datasets. *IEEE Access*, 10:118454–118464, 2022.
- [26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
- [27] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [28] MathWorks. What is machine learning? <https://www.mathworks.com/discovery/machine-learning.html>, 2022.

- [29] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR)*. [Internet], 9:381–386, 2020.
- [30] Rene Y Choi, Aaron S Coyner, Jayashree Kalpathy-Cramer, Michael F Chiang, and J Peter Campbell. Introduction to machine learning, neural networks, and deep learning. *Translational Vision Science & Technology*, 9(2):14–14, 2020.
- [31] Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018.
- [32] AMAZON AWS. What is a neural network? <https://aws.amazon.com/what-is/neural-network/>.
- [33] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *Towards Data Sci*, 6(12):310–316, 2017.
- [34] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, 3(3):196–207, 2020.
- [35] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 2022.
- [36] AnalyticsVidhya. Activation functions and their derivatives – a quick complete guide. <https://www.analyticsvidhya.com/blog/2021/04/activation-functions-and-their-derivatives-a-quick-complete-guide/>.
- [37] Softmax function. <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer>.
- [38] IBM. What are convolutional neural networks? <https://www.ibm.com/topics/convolutional-neural-networks>.
- [39] Kuniyiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

- [40] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.
- [41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [42] Francisco Erivaldo Fernandes Junior and Gary G Yen. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm and Evolutionary Computation*, 49:62–74, 2019.
- [43] Arun Kumar Dubey and Vanita Jain. Comparative study of convolution neural network’s relu and leaky-relu activation functions. In *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*, pages 873–880. Springer, 2019.
- [44] Mduduzi Manana, Chunling Tu, and Pius Adewale Owolawi. A survey on vehicle detection based on convolution neural networks. In *2017 3rd IEEE international conference on computer and communications (ICCC)*, pages 1751–1755. IEEE, 2017.
- [45] Wikipedia. Kernel(image processing). [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing)).
- [46] Sabina Pokhrel (Towards Data Science). Beginners guide to convolutional neural networks. <https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d>.
- [47] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.
- [48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.
- [49] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

- [50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [51] What is computer vision? <https://www.ibm.com/topics/computer-vision>. access date: 07-04-2023.
- [52] Suman Paneru and Idris Jeelani. Computer vision applications in construction: Current state, opportunities & challenges. *Automation in Construction*, 132:103940, 2021.
- [53] M Aharchi and M Ait Kbir. A review on 3d reconstruction techniques from 2d images. In *Innovations in Smart Cities Applications Edition 3: The Proceedings of the 4th International Conference on Smart City Applications 4*, pages 510–522. Springer, 2020.
- [54] Toward Data Science. Object detection with convolutional neural networks. <https://towardsdatascience.com/object-detection-with-convolutional-neural-networks-c9d729eedc18>.
- [55] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.
- [56] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.
- [57] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.
- [58] Aditya Lohia, Kalyani Dhananjay Kadam, Rahul Raghvendra Joshi, and Anupkumar M Bongale. Bibliometric analysis of one-stage and two-stage object detection. *Libr. Philos. Pract*, 4910:34, 2021.
- [59] Medium(Sieun Park). A guide to two-stage object detection: R-cnn, fpn, mask r-cnn. <https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c>.



- [60] Abhishek Balasubramaniam and Sudeep Pasricha. Object detection in autonomous vehicles: Status and open challenges. *arXiv preprint arXiv:2201.07706*, 2022.
- [61] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [62] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104:154–171, 2013.
- [63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.
- [64] Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1458–1465. IEEE, 2005.
- [65] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [66] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [67] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [68] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [69] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

- [70] Medium(Jonathan Hui). Ssd object detection: Single shot multibox detector for real-time processing. <https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c>.
- [71] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.
- [72] V7labs. The beginner’s guide to self-supervised learning. <https://www.v7labs.com/blog/self-supervised-learning-guide>.
- [73] NeptuneAi. Self-supervised learning and its applications. <https://neptune.ai/blog/self-supervised-learning>.
- [74] Meta Ai. Self-supervised learning: The dark matter of intelligence. <https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/>.
- [75] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [76] Amit Chaudhary. The illustrated simclr framework. <https://amitnness.com/2020/03/illustrated-simclr>, 2020.
- [77] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *Advances in neural information processing systems*, 33:6827–6839, 2020.
- [78] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.
- [79] Paul Xiong. Compare simclr, byol, and swav for self-supervised learning (1). <https://paulxiong.medium.com/compare-simclr-byol-and-swav-for-self-supervised-learning-1-3e43e5c4e8cb>.

- [80] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.
- [81] Shilpa Ananth. Faster r-cnn for object detection. <https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46>.
- [82] Daniel Godoy. Understanding binary cross-entropy / log loss: a visual explanation. <https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>.
- [83] Chien-Yao Wang, Hong-yuan Liao, and I-Hau Yeh. Designing network design strategies through gradient path analysis. 11 2022.
- [84] Ali Bahari Malayeri and Mohammad Bagher Khodabakhshi. Concatenated convolutional neural network model for cuffless blood pressure estimation using fuzzy recurrence properties of photoplethysmogram signals. *Scientific Reports*, 12(1):6633, 2022.
- [85] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.
- [86] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. Learning non-maximum suppression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4507–4515, 2017.
- [87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [88] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016.
- [89] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3286–3295, 2019.

- [90] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [91] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. Seed: Self-supervised distillation for visual representation. *arXiv preprint arXiv:2101.04731*, 2021.
- [92] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.
- [93] Difference between a batch and an epoch in a neural network. <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/>.
- [94] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.