Equipment identification through image recognition

Saidnassimov Darkhan

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 29.7.2022

Supervisor

Prof. Alexander Ilin

Advisor

Dr. Christian Binder

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Saidnassimov Darkhan

Title Equipment identification through image recognition

Degree programme Automation and Electrical Engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Alexander Ilin

Advisor Dr. Christian Binder

Date 29.7.2022 **Number of pages** 94+2 **Language** English

Abstract

Object detection is a rapidly-evolving field with applications varying from medicine to self-driving vehicles. As the performance of the deep learning algorithms grow exponentially, countless object detection applications have emerged. Despite the nearly all-time high demand, object detection is rarely used in industrial applications. Historically, object detection requires extensive training data in order to produce sufficient results. Collecting huge datasets is often impractical in an industrial environment due to the confidentiality restrictions and data accessibility limitations.

This thesis attempts to minimize the manual labeling process by proposing a regularized cross-domain adaptive teacher model with continual learning. The model assumes a task that seeks to eliminate the domain shift between industrial datasets: a larger labeled dataset of rendered images and a smaller unlabeled dataset of real-life images. While the labels for the rendered images can be generated automatically, only a tiny amount of real images needs to be collected, which is crucial for the system scalability in industrial environments. The model transfers knowledge from one domain to another by means of adversarial domain adaptation and mean teacher training. In an attempt to achieve state-of-the-art results, this thesis proposes to regularize the student and the teacher networks using image-and instance-level alignment as well as consistency loss. Additionally, the model adopts a lifelong learning approach with network expansion and gradient regularization that enables the model to be retrainable on a continuously expanding dataset, which further facilitates the scalablity of the system.

As a result, the proposed Adaptive teacher model with two-level alignment achieved competitive results with `AP50` = 69.57 % at 14 999 iterations, which is twice as fast compared to the original model with `AP50` = 71.40 % at 30 999 iterations. On the other hand, the continual learning experiment with 10 arbitrary classes proved that retraining the model on the entire dataset `AP50`) = 63.72 % brings more benefit than training the model continuously using the proposed approach `AP50`) = 45.56 %. Finally, the proposed model was evaluated on one Metso Outotec equipment item, which included 1000 labeled rendered images and 28 unlabeled real images. The tests achieved a fair performance of `AP50` = 85.86 %.

Keywords computer vision, object detection, transfer learning, domain adaptation, cross-domain object detection, continual learning

Preface

I would like to thank Professor Alexander Ilin at Aalto University for his excellent guidance. Additionally, I would like to thank Dr. Christian Binder for offering the opportunity at Metso Outotec and providing full support throughout the process. I would also like to thank my colleagues that motivated me endlessly during my internship. Finally, I would like to thank the CSC Finnish IT center for the computing resources that made the research possible.

Otaniemi, 29.7.2022

Saidnassimov, D.

Contents

List of Figures

List of Tables

Symbols and abbreviations

Symbols

λ	Trade-off parameter
$\mathcal{L}$	Loss function
$\mathcal{W}$	Weights matrix
$\mathcal{D}$	Domain
$\mathcal{D}_\mathcal{S}$	Source domain dataset
$\mathcal{D}_\mathcal{T}$	Target domain dataset
$\mathcal{X}$	Feature space
$\mathcal{X}_\mathcal{S}$	Feature space of the source domain
$\mathcal{X}_\mathcal{T}$	Feature space of the target domain
$\mathcal{F}$	Feature vector
$\mathcal{Y}$	Label space
$\mathcal{Y}_\mathcal{S}$	Label space of the source domain
$\mathcal{Y}_\mathcal{T}$	Label space of the target domain
$P(\mathcal{X})$	Marginal probability distribution of $\mathcal{X}$
$P(\mathcal{X}, \mathcal{Y})$	Joint distribution
$P(\mathcal{Y} \mid \mathcal{X})$	Conditional distribution
$\mathcal{N}$	Number of samples
$\mathcal{D}_\mathcal{S} = \{\mathcal{X}_\mathcal{S}^i, \mathcal{Y}_\mathcal{S}^i\}_{i=1}^{\mathcal{N}_\mathcal{S}}$	Labeled source domain
$\mathcal{D}_\mathcal{T} = \left\{\mathcal{X}_\mathcal{T}^j\right\}_{j=1}^{\mathcal{N}_\mathcal{T}}$	Unlabeled target domain
$\mathcal{T}$	Task
$\mathcal{T}_n$	Task of iteration n

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	derivative with respect to variable t
$\frac{\partial}{\partial t}$	partial derivative with respect to variable t
$\sum_{i=1}^{n}$	sum over index i until n
$A \cap B$	the intersection of two sets or areas
$A \cup B$	the union of two sets or areas

List of Abbreviations

AI	Artificial intelligence
ML	Machine Learning
DL	Deep Learning
GPU	Graphical Processing Unit
CPU	Central Processing Unit
ANN	Artificial Neural Network
DNN	Deep Neural Network
FC	Fully-Connected (layer)
CNN	Convolutional Neural Network
RCNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
MSE	Mean-Squared Error
LR	Learning Rate
SGD	Stochastic Gradient Descent
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning
VOC	Visual Object Classes
COCO	Common Objects in Context
ResNet	Residual Neural Network
RPN	Region Proposal Network
RCNN	Regions with CNN features
ROI	Region of Interest
FPS	Frames Per Second
YOLO	You Only Look Once
SSD	Single-shot MultiBox Detector
IoU	Intersection over Union
AP	Average Precision
mAP	Mean Average Precision
NMS	Non-Maxumum Suppression
TL	Transfer Learning
DA	Domain Adaptation
UDA	Unsupervised Domain Adaptation
DANN	Domain Adversarial Neural Network
GAN	Generative Adversarial Network
T-LESS	Texture-LESS
CAD	Computer-Aided-Design
API	Application Programming Interface
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

1 Introduction

1.1 Problem statement

In recent years, computer vision algorithms have received much attention due to their potential applications in a vast variety of fields, including security monitoring [1], medicine [2], and self-driving vehicles [3, 4]. However, although computer vision has been integrated into industrial applications (e.g., safety and process monitoring) [1, 5], less research has addressed the issue of industrial equipment detection [6, 7, 8].

As industrial plants are typically in the range of hundreds of meters in size, it often becomes challenging to manually recognize equipment parts for maintenance or replacement. Ore processing plants treat several hundred tons of ore per hour, and the production capacity is constant. Therefore, it is often difficult to properly identify the equipment within a list of thousands of parts in a medium- to large-scale plant.

This work has been commissioned by Metso Outotec Oyj. Metso Outotec offers digital solutions that enable customers to automate their processes in the mining, aggregates and metals industries. In order to ensure that these processes operate as smoothly as possible, it is important to optimize them at all stages of production. Recently, Metso Outotec has successfully applied computer vision in applications for identifying foreign objects in crushing processes [9], for detecting defects in copper molds [10], as well as for recognizing froth characteristics in flotation cells [11], to name a few. However, the company has not yet attempted to apply computer vision for facilitating maintenance. For these reasons, Metso Outotec has requested to investigate the feasibility of applying state-of-the-art computer vision algorithms to equipment recognition.

Even though various methods have been implemented for detection of objects in a countless number of fields [12, 13, 14, 15, 16, 17], these methods heavily rely on extensive data collection and training of models in order to accurately identify objects. Moreover, complications arise, as it is often not possible to collect huge amounts of training images from industrial environments due to privacy and confidentiality issues. Luckily, for this project, the images can rather easily be collected from a 3D simulator model of a metal refining plant. However, using the rendered images from a 3D simulator limits the accuracy of the models, as such models do not perform as well on real images due to the domain shift phenomenon [18], which occurs when the environmental conditions change at the time of capturing training and test images.

Hence, this thesis proposes a cross-domain object detection approach as a solution to automatically localize and identify the equipment in a large industrial environment in order to minimize the delay in production arising as a result of manual identification.

1.2 Thesis objective

The main goal of this thesis is to identify a suitable state-of-the-art object detection technique and to enhance its performance on the custom equipment dataset. The proposed method should be able to identify an object in a real image given a labeled dataset of rendered images from a 3D model and a smaller unlabeled dataset of real images. Additionally, the developed method should provide a solution for optimizing the laborious process of data collection and labeling. Furthermore, the produced model should address the cases when new objects are continuously added to the dataset. Such optimization is important not only because training the model from the scratch is a time-demanding process, but also because large plants contain thousands of objects, thus making scalability a critical requirement. Finally, a minimal proof-of-concept application should be prepared to demonstrate the performance of the proposed detection technique on one equipment item from an industrial plant.

1.3 Methodology

In order to accomplish these objectives, the thesis will first explore state-of-the-art object detection frameworks, libraries and algorithms. Similarly, domain adaptation algorithms will be analyzed in an object detection setup. The most suitable methodologies will then be selected to be used in a novel cross-domain object detection model.

In order to circumvent regulations regarding accessibility and confidentiality, the dataset utilized for training the model in the experimental scenario will be primarily based on the T-LESS open-source dataset [19]. Since the dataset was originally intended for pose estimation in 3D models, it will be converted into formats appropriate for the proposed object detection algorithms.

To achieve higher performance in object detection, the domain shift phenomenon will be addressed using the Adaptive Teacher [20] algorithm for cross-domain object detection, which in turn uses the Faster-RCNN [12] implementation as a detector base in the Detectron2 [21] framework. The thesis will contribute to current knowledge by introducing an instance-level domain classifier appended to the base network of the Adaptive Teacher algorithm, as suggested by Chen et al. [22]. Additionally, the study will evaluate the feasibility of other strategies, such as continual learning [23], to further enhance scalability of the model. The model will then be integrated into a prototype web application for demonstration purposes. The produced model will be trained on rendered data from 3D models and evaluated on real images using mean average precision metrics. Finally, the proposed method will be evaluated using one equipment item from a real plant operated by a Metso Outotec client.

1.4 Scope

The thesis will be limited to proposing a minimal proof-of-concept solution based on analyzing and combining different components of existing state-of-the-art models. In addition, this solution will be wrapped in a prototype web application. However, preparing an actual real-life dataset and implementing the solution for a real plant remains outside the scope of this study due to the time constraints. Although the proposed method attempts to optimize the data collection and labeling process, this will in practice require many months before the dataset and the model based on real data would be ready for use.

For the user interface, a prototype will be provided in order to showcase the performance of the model. However, the thesis will primarily focus on deep learning algorithms rather than methods to deploy a model. For this reason, the prototype will only offer basic functionality. Finally, due to time constraints, a video-compatible model will remain outside the scope of this work.

1.5 Structure of the thesis

The rest of this thesis is divided into four chapters. Chapter 2 reviews the common terminology on deep learning, image classification and object detection. Additionally, the chapter discusses the state-of-the art research in industrial object detectors. Furthermore, the latest domain adaptation and cross-domain object detection techniques are covered in greater detail. Finally, the topic of continual learning is reviewed. Chapter 3 defines the dataset, the evaluation metrics used, as well as evaluates the standard object detectors and several cross-domain object detection algorithms. Finally, it proposes a novel architecture, which addresses both domain adaptation and continual learning. Chapter 4 evaluates the proposed solution and compares the results to other methods using average precision metrics. Additionally, the proposed solution is evaluated on one equipment item. Chapter 5 summarizes this work by discussing the proposed architecture and suggesting directions for future work.

2 Background

This section of the thesis introduces the key concepts related to the field of study. The section mainly discusses neural networks, object detection, transfer learning and domain adaptation. Additionally, the section will familiarize the reader with the relevant terminology and notations used. Furthermore, this section will briefly introduce the state-of-the-art industrial object detectors. The section will provide an extensive overview of the latest domain-adaptive object detection methods. Finally, the topic of continual learning will be introduced.

2.1 Deep learning and neural networks

Historically, machine learning (ML), a sub-field of Artificial Intelligence (AI), has been a highly computational task. The primary cause of it was linked to low hardware performance. According to Moore's law [24], the amount of transistors doubles in a circuit in a given number of months. As the computational power of the computers grew proportionally to the number of transistors, the results have been steadily improving. The improvement was further facilitated with the discovery of the Graphical Processing Unit(GPU) applicability in ML tasks [25]. Additional critical bottlenecks in ML were caused by sub-optimal algorithms and data availability limitations. As the availability of data improved, new fields of applications arose. These and many other advancements made it possible to accelerate the training speed of deep neural networks(DNN).

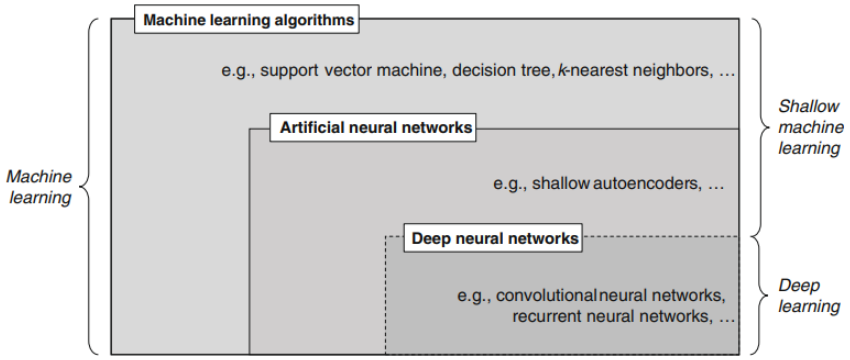


Figure 1: Machine learning concepts [26].

The concept of deep neural networks originates from biology, where a network of neurons is fundamental to the functionality of a brain. In overly simplified terms, such network consists of interconnected neurons that capture an external signal and produce a certain reaction within the brain as a response. Figure 2 (a) illustrates a typical neuron, where the signal flows from dendrites through the cell-body of the neuron. If the signal is strong enough, the neuron activated and passes the signal

further to other neurons through the connections called "synapses", as shown in Figure 2 (c). Identical process takes place in remaining neurons, which ultimately forms a neural network [27].

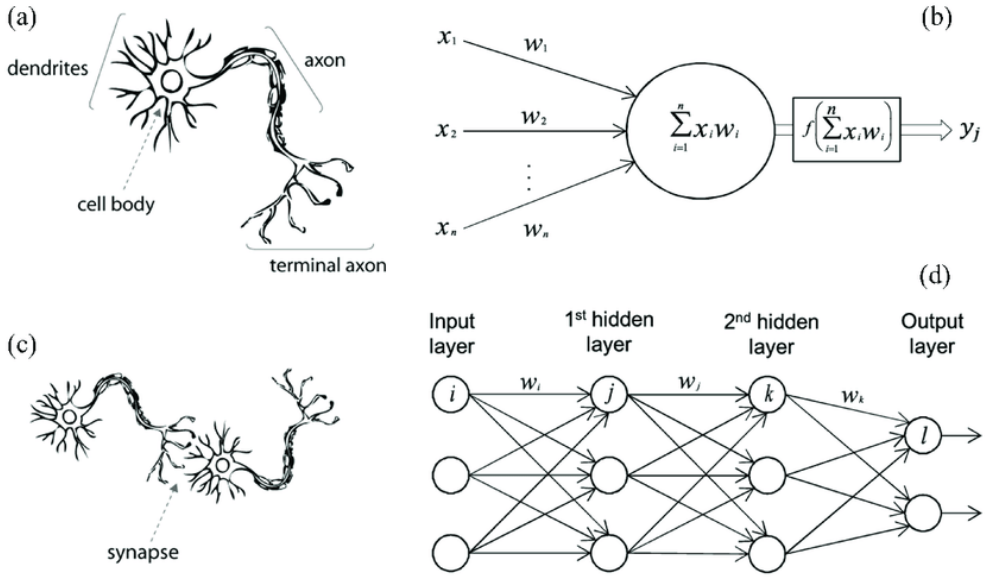


Figure 2: A biological(a) neuron against artificial(b) and biological synapse(c) against artificial(d) [28].

In deep learning(DL), a sub-field of ML, this architecture has been borrowed to implement an artificial neural network(ANN), where a neuron is simply a unit that processes an input signal. Figure 2 (b) demonstrates a simplified structure of an artificial neuron. Here, $x_1, x_2..x_n$ represent input signals, while $\mathcal{W} = w_1, w_2, ...w_n$ are the weights of the signal and y_i is the output of the neuron. The higher the weights of the input are, the stronger the influence of the neuron on the output. The weighted sum of the inputs is then passed to the activation function, which essentially determines the output of the node and ultimately allows to learn complex patterns in data [27].

A few of the most popular non-linear activation functions include a logistic sigmoid, tanh function, softmax and a rectified linear unit(ReLU). Among the four, ReLU has been considered state-of-the-art in the field of deep learning due to the performance in convolutional neural networks(CNN) [29] and the simplicity. The logic of ReLU can be represented as follows:

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Consequently, the output of the activation function is passed to a hidden layer of

neurons, as illustrated in Figure 2 (d). The layers in the middle are called "hidden" due to the fact that both outputs and inputs are masked by the activation function. The hidden layers will calculate the weighted output of the previous layers until the signal eventually reaches the final output layer of the network. Hidden layers that stack up together to form a classical deep learning architecture [30]. Such architecture allows to process data in a non-linear pattern. In the original ANN all the layers are fully-connected(FC), meaning that each node of the input vector affects each node of the output vector, as shown in Figure 2 (d).

Due to the biological nature, neural networks adapt over time by creating new connections between neurons. The neurons in ANN adopted such behaviour by utilizing a backpropogation algorithm [31]. A naive backpropogation approach is illustrated in Figure 3. The algorithm consists of two parts: feed-forward and backward loops. Generally, the main objective of an ANN is to choose such weights that the network produces desired target outputs. The forward pass propagates along the nodes at each layer of the network and returns a predicted output. A cost (loss) function is used to compare the ground-truth output to the prediction to evaluate the quality of the weights in the network. The classic example of a cost function is the mean-squared error (MSE). MSE is often used in regression based problems and its principle is shown in Equation 2.

$$\text{argmin}\frac{1}{n}\sum_{i=1}^{n}\left(f\left(x_i\right)-y_i\right)^2 \tag{2}$$

The MSE cost function attempts to minimize the difference between the prediction and the target output, while giving more weight to larger distances due to the squared output [32].

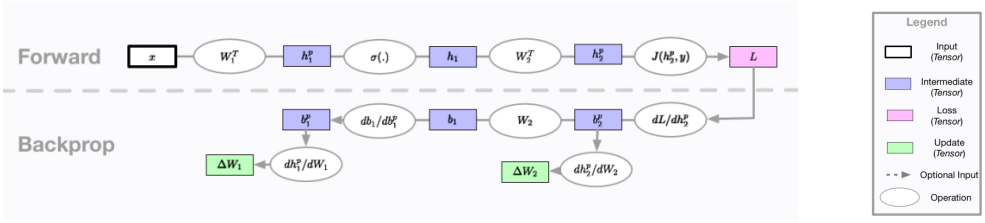


Figure 3: Backpropogation algorithm, adapted from [33].

To minimize the loss function, the algorithm calculates a partial derivative of the loss term $\mathcal{L}$ with respect to the weights: $\nabla g = \frac{\partial \mathcal{L}(f(x_i),y_i)}{\partial \mathcal{W}}$. The basic algorithm of the gradient descent is commonly used in optimizing such functions and its logic can be generalized as follows:

1. Initialize the algorithm with a random value of θ^i, where starts with $i = 0$.
2. Define the next θ value as $\theta^{i+1} = \theta^i - \eta\left(\left(\nabla g\right)\left(\theta^i\right)\right)$.
3. Iterate over the values of i and repeat until convergence [32].

Generally speaking, three gradient descent algorithms are most commonly used. These algorithms include batch gradient descent, stochastic gradient descent (SGD) and mini-batch gradient descent. While the batch gradient descent updates the model only after all the samples have been evaluated, the SGD calculates the error for one sample in the dataset and updates the parameters one by one. Finally, the mini-batch gradient descent algorithm divides the data into smaller batches and calculates updates on each of the data subsets.

Finally, by applying the chain rule to the derivatives in a backwards-direction, the updates for the matrix $\mathcal{W}$ are calculated for the nodes in the neural network [33]. The algorithm is then repeated until the global minimum is reached. The process of determining the weight values to utilize in each subsequent layer in the neural network by means of backpropogation algorithm is called "training the model". During training, the value that ΔW are updated is known as "learning rate" (LR). This hyperparameter of the network typically equals to a small value between 0 and 1. Picking too high LR value would result in overshooting e.g. never finding the global minimum, while too low value will lead to slow convergence and higher computational costs.

2.2 Neural networks in computer vision

With the discovery of DNN, many of the popular computer vision techniques became obsolete. Specifically, the introduction of CNNs was an important milestone in boosting machine perception performance [34]. Nowadays, CNNs are typically used to address various pattern recognition and computer vision tasks. Some of the tasks include:

- Image Classification
- Object Detection
- Segmentation
- Facial Recognition
- Domain Adaptation
- Image Reconstruction
- and many others [35]

For addressing the key objectives of this work (Section 1.2), the thesis will extensively cover image classification, object detection and domain adaptation tasks. Figure 4 illustrates a simplistic CNN architecture approach to the MNIST [36] classification problem. A CNN is essentially a neural network that leverages convolutional layers to produce predictions. Unlike the traditional computer vision methods, CNNs do not need to extract features of the image beforehand due to the logic behind convolution. In classical ML the features are extracted separately, followed by the

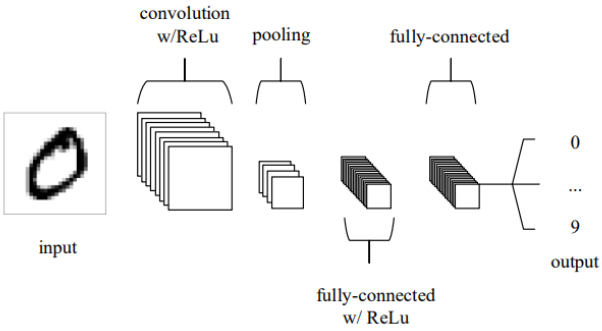


Figure 4: A simple CNN network with 5 layers for image classification task [34].

appropriate algorithms for learning. On the contrary, DL algorithms, such as CNN, learn the features automatically [37].

To understand the logic behind convolutional layers, it is important to discuss the operation of convolution. Convolution is a mathematical operation of two functions that indicates how the shape of one is affected by another. In terms of image processing, convolution is a process, where the kernel moves along the input matrix dimensions. Each output pixel can then be calculated as the dot product of the cropped input and the kernel [38]. Figure 5 (a) illustrates the process of convolution in image processing.

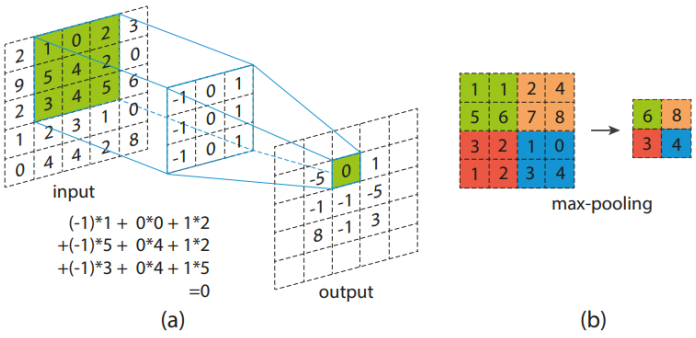


Figure 5: The process of convolution [38].

As a result, the elements of the network are not densely connected, which allows better generalization and flexibility. This operation in practice allows to extract the important features of the image, such as edges, corners, shapes and many others.

However, unlike in the FC layers, the number of weights is much smaller, which is essential when it comes to high-dimensional images. Following the architecture in Figure 4, the outputs of the convolutional layers are activated with ReLU in a similar manner as in a classical ANN structure. The outputs are then pooled in order to down-sample the image and hence reduce the computational costs [38]. The commonly used max-pooling operation is shown in Figure 5 (b). Max-pooling down-samples the image by applying a filter that extracts the highest value in the region of the feature map. Finally, fully-connected layers complete the structure of the basic CNN. In order to flatten the outputs from the convolutional layers, the network is finalized with the FC layers. In case of the input in Figure 4, this in turn allowed to calculate the probability of the input image to represent a number between 0 and 9 [34].

CNNs boosted the performance in computer vision tasks. However, deep learning methods still required extensive training data. Luckily, multiple algorithms emerged as large datasets became available, which were made public as different image classification challenges appeared. The datasets commonly used in benchmarking are ImageNet [39], PASCAL Visual Object Classes (VOC) [40] and Common Objects in Context (COCO) [41]. The algorithms that emerged as a result of these challenges will be discussed in the following section.

2.3 Image classification

It is important to understand the concepts of image classification before moving on to object detection principles. Among the three mentioned earlier, ImageNet was chosen to be a de-facto dataset for running benchmarks. Different CNN-based models were proposed and some of the most popular models include LeNet [42], AlexNet [43], VGGNet [44] and Residual Neural Network(ResNet) [45]. As it can be concluded from Figure 6, the classification error dropped lower than the error of the manual detection with the introduction of ResNet, thus approaching the theoretical limits.

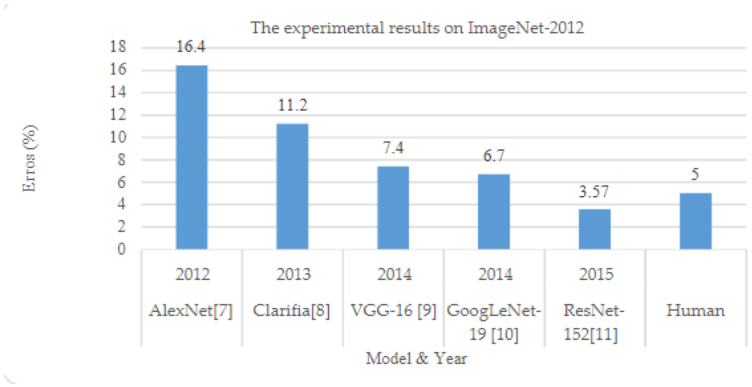


Figure 6: Evolution of image classifier models evaluated on ImageNet dataset [37].

2.3.1 LeNet

LeNet architecture (1998) [42] is considered to be a pioneer in the field. Its design inherited the classic CNN architecture, but instead it consisted of 7 layers, as presented in Figure 7. However, the implementation of the paper was not possible for more than 10 years due to limitations in computing power at the time.

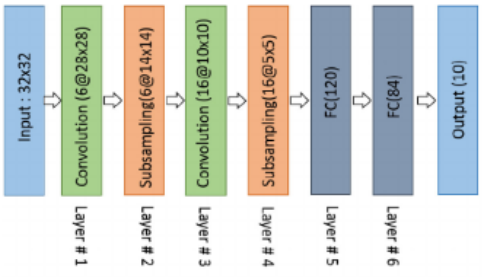


Figure 7: LeNet architecture [37].

2.3.2 AlexNet

Consequently, AlexNet paper was introduced, which proved the effectiveness of their model, as it outperformed the leading implementations of that time and achieved the error rate of 15.3% [43]. Figure 8 displays the proposed network. The architecture of AlexNet is similar to one of LeNet, though it is substantially deeper and has more than 60 million adjustable parameters. It has 5 convolutional layers of varying kernel size. The convolutional layers are activated using ReLU the operation of max-pooling is additionally applied. The architecture is finalized by attaching two FC layers with dropout rate of 0.5 and a softmax layer.

During dropout, there is a probability that the neuron will be excluded from computations in the subsequent layer. Utilizing such technique proved to be essential to fight over-fitting in FC layers and improve generalization [46].

AlexNet architecture was the first CNN to split the model into two parts and to leverage mulitple GPUs in training due to GPU memory limitations of 3GB at the time [43].

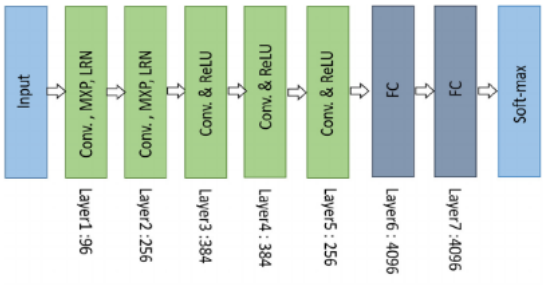


Figure 8: AlexNet architecture [37].

2.3.3 VGG

Another milestone was achieved with the discovery proposed in VGGNet [44]. This work proved that increasing the depth of the network greatly impacts on the performance of the CNN in classification tasks [37]. Three different versions of the model were proposed with 11, 16 and 19 layers, respectively and with the deeper model being the best in performance, but more expensive in terms of computation. Equivalently to AlexNet, the network has blocks of convolutional layers of a mixed kernel size, followed by a ReLU block and max-pooling. The network is finalized with three FC and one softmax layers.

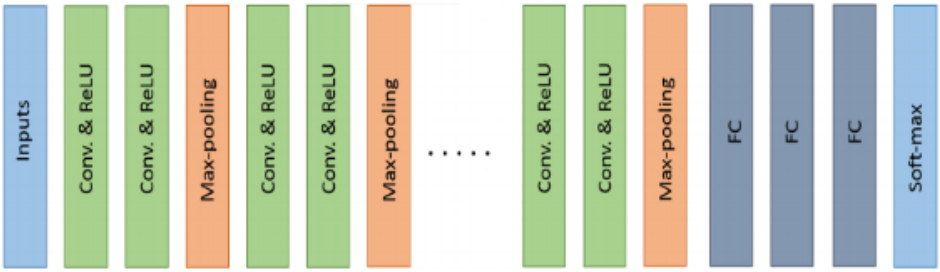


Figure 9: VGG architecture [37].

2.3.4 ResNet

ResNet architecture [45], which was proposed in 2015, discovered that after reaching certain depth of the network, the performance of the model degrades. He et al. [45] suggested that this happens due to the "vanishing gradient" problem. As the model gets deeper, several applications of the chain rule on during backpropogation tend

to diminish either all the way to zero or becomes too large. As a result, no update is applied on the weights and hence, no training takes place. He et al. proposed to utilize a residual block, illustrated in Figure 10, which is used to skip some of the layers in between. This solution essentially mitigates the problem of vanishing gradients by allowing the training loop to skip parts of the network that negatively affect the performance.

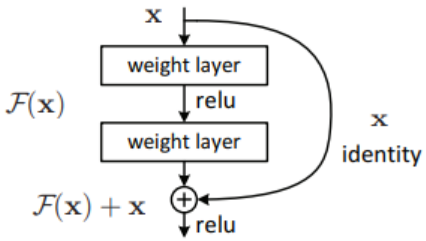


Figure 10: Residual block of ResNet [45].

ResNet adopts the VGG-19 architecture, but adds a skip connection block. He et al. implemented multiple versions of the model and the largest one is 152 layers deep. Nonetheless, the network has less trainable parameters than VGG, which substantially improves the training speed while preserving accuracy due to the residual block. As a result, the proposed model achieved the error of 3.57% on ImageNet dataset, which is lower than the human eye error [45].

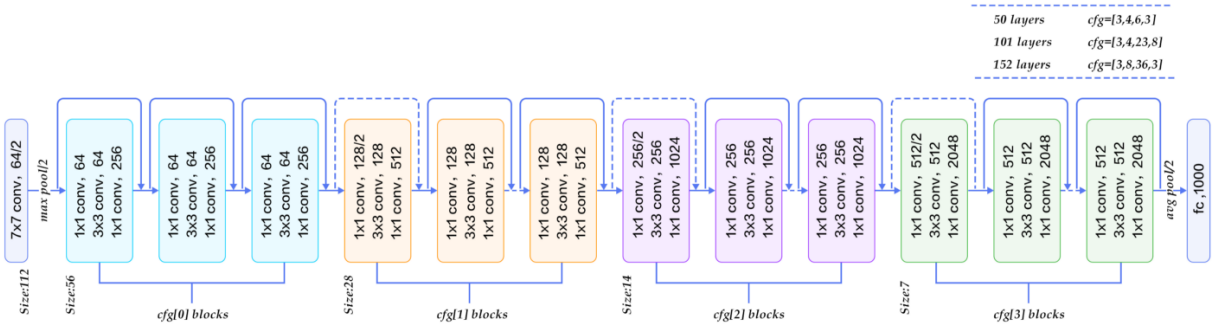


Figure 11: ResNet architecture [47].

2.4 Object detection

Object detection is an extended version of the image classification problem. However, unlike image classification, object detection aims to recognize not only the object, but also to localize it. Prior to arrival of deep learning, object detection considered to be a difficult task. With the discovery of the CNN architecture, the traditional computer vision techniques became obsolete. Since then, multiple different detector algorithms emerged. A typical object detection network contains two important modules - the base (or backbone) network and the detector network. A base network is usually one of the pre-trained VGG or ResNet models, presented in the previous chapter. A base network extracts the features, which are then passed to a detector. Generally speaking, state-of-the-art detector networks can be classified into two categories: single- and two-stage detectors [48], as shown in Figure 12.

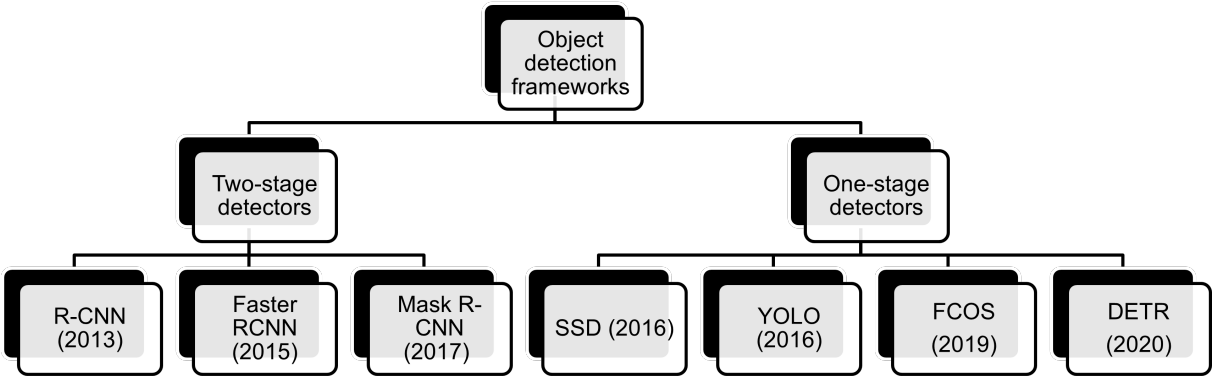


Figure 12: Types of object detectors.

Two-stage object detectors attempt to generate regions of an image that contain an object first, and then performs the classification and localization tasks in the proposed area. On the other hand, single-stage detectors try to detect objects directly without running the Region Proposal Network (RPN). Figure 13 illustrates the key differences in the architecture.

Typical examples of the single-stage networks are RetinaNet [50], Single-Shot Detector(SSD) [13] and You Only Look Once(YOLO) [15]. On the contrary, Region-based CNN(R-CNN) [51], Fast-RCNN [52] and Faster-RCNN [12] are considered to be the most important two-stage detectors. Figure 12 also mentions detectors such as FCOS [17], Mask-RCNN [14] and DETR [53]. Although they show competitive performance, they are not reviewed in the thesis as they are not relevant to the method introduced in the Research Methodology. The following subsection will introduce the reader to some of the main concepts of the selected detectors.

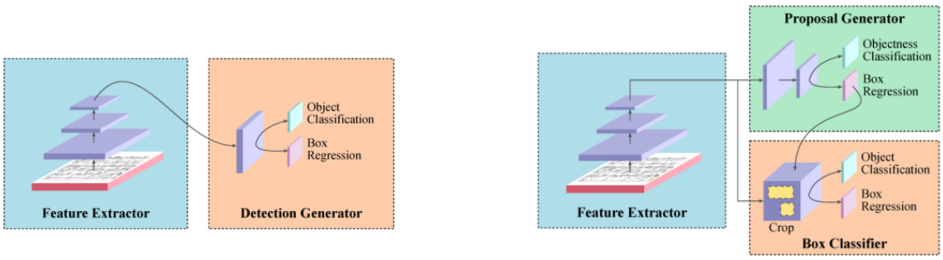


Figure 13: A simple single-stage detector(left) compared to a two-stage detector(right) [49].

2.4.1 R-CNN

Figure 14 shows the structure of the R-CNN network [51]. R-CNN has been one of the first CNN-based models to be introduced [48]. Girshick et al. essentially suggested to use a module that extracts the object proposals and then to pass it to a CNN. The CNN would then extract the features relevant, which would in turn allow to classify the proposed region as well as to localize it. In their original experiments, the selective search algorithm [54] was used to produce roughly 2000 regions. AlexNet [43] was used as a backbone CNN to extract vectors of 4096 size dimensions. The features were then passed to binary classifiers that are bound to a certain class. As multiple regions are returned, the Non-Maximum Suppression(NMS) algorithm [55] is used to identify the best proposal for the object.

Unfortunately, the inference process in R-CNN took a whole 47 seconds per image [51], which was not fast enough to be remain relevant in the object detection field.

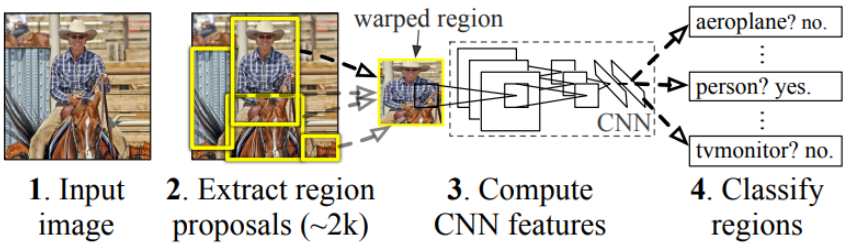


Figure 14: R-CNN overview [51].

2.4.2 Fast-RCNN

Unlike the classic R-CNN system, where the components such as the classifier and the regressor have to be trained separately, the same authors of the RCNN model also introduced a Fast-RCNN model [52]. Fast-RCNN incorporated an end-to-end

trainable system. Additionally, the Region-of-Interest (ROI) layer was proposed. This component essentially splits the image into a fixed-size grid and applies average pooling. The resulted one-level pyramid of the image is used as a feature map for the detector head. This allows the rest of the network to focus purely on the features extracted for the proposed regions. Finally, two additional FC output layers are appended, where the first one classifies the features by returning the probability of N object classes in range $(0, N + 1]$, which includes the background class as well. The other one returns a vector of a size $(0, 4 \times N]$, thus including the bounding box coordinates for the foreground classes only [52]. Fast-RCNN showed a significant($\times$146) speed improvement as compared to R-CNN, thus allowing it to be applied in real-time [51]. The original architecture of Fast-RCNN is shown in Figure 15 below.

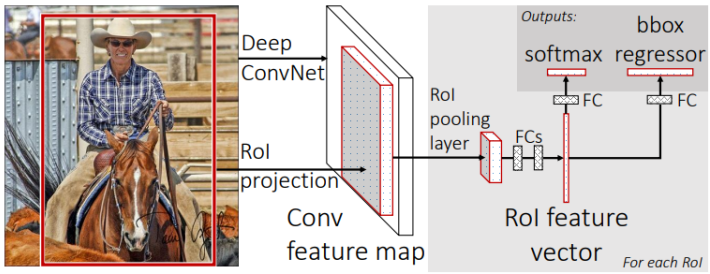


Figure 15: Fast-RCNN overview [52].

2.4.3 Faster-RCNN

Ren et al. [12] discovered that even though Fast-RCNN was significantly faster than the preceding networks, the region proposing process was still causing a bottleneck. Therefore, in the novel Faster-RCNN [12] architecture it was suggested to replace the legacy region proposal module with a fully-convoluted architecture called region proposal network(RPN) [51]. Instead of using a one-level image pyramid to address the object detection problem as it was proposed by Girshick et al. [52], Faster-RCNN utilizes anchor boxes of different aspect ratios to propose object candidates. Similarly to any other CNN, the features in Faster-RCNN are extracted from the convolutional layers based on a VGG backbone. The features maps are then sent to the RPN, which in practice is a sliding window of $n \times n$ $(n = 3)$ dimensions. At every sliding window position, the k number of object proposals are predicted. Such boxes are called "anchors" as illustrated in Figure 16. The acquired $2k$ classification predictions whether the proposed region is an object of interest or not and 4 regression outputs of the bounding box coordinates are then mapped back to the ROI layer and, eventually, to the FC layers that predict the class of the proposed object [12].

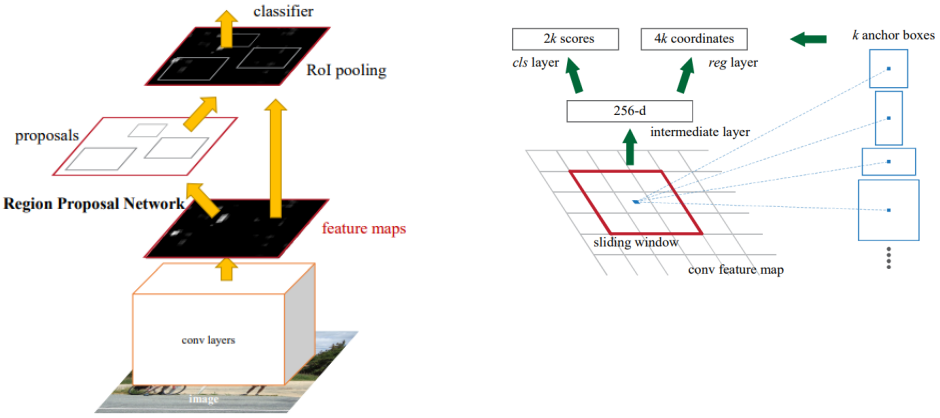


Figure 16: Faster-RCNN overview and its RPN module [12].

As the network returns the predictions, the loss function is as follows:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right) \ , \tag{3}$$

where p_i is the probability of the predicted anchor i to be an object, p_i^* is a ground-truth label from 0 to 1, t_i is a vector with 4 corner coordinates of the bounding box and t_i^* is its corresponding ground-truth vector. λ is a trade-off between the L_{cls} and L_{reg}, where L_{cls} is calculated as a binary cross-entropy loss and L_{reg} is a smooth L_1 loss that was introduced in Fast-RCNN paper [52].

This implementation, unlike Fast-RCNN, allowed to return predictions nearly in real time with the breakthrough of 5 frames per second(FPS) [12]. Although the two-stage Faster-RCNN detector is nearly 7 years old, it is still one of the most widely used detectors in the field, as can be noticed from Figure 17. In later chapters, this work will focus on the Faster-RCNN implementation. Nevertheless, it is worth mentioning single-stage competitors. Often they are slightly weaker in terms of accuracy, compared to two-stage detectors. However, these detectors offer a significant improvement in robustness as compared even to the fastest one of the two-stage detectors, Faster-RCNN.

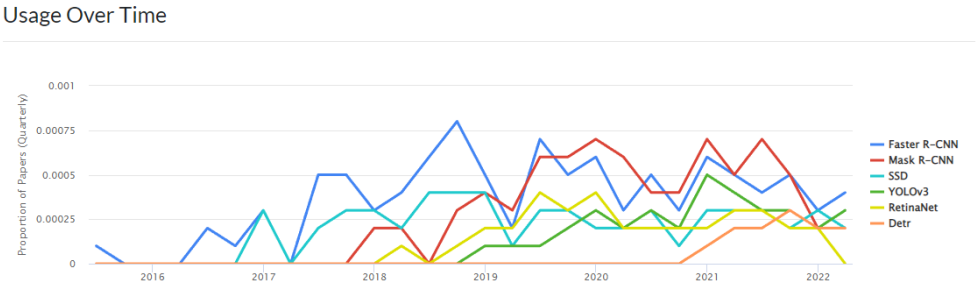


Figure 17: The popularity of different detectors according to [56].

2.4.4 YOLO

Unlike the two-stage methods presented, You Only Look Once(YOLO) [15] algorithm addresses object detection as a purely regression problem by predicting the bounding boxes of the objects directly without region proposals. The YOLO network instead splits the image into a grid of $S \times S$ cells, for which a B number of bounding boxes and C probabilities of the class and are predicted [15]. The overview of the detection process is shown in Figure 18. The principle of the YOLO grid component is broadly similar the one of R-CNN [51], where the algorithm of selective search [54] is used to propose regions. However, instead of proposing more than 2000 regions, YOLO only returns 98 proposal boxes per image. This, along with having an optimized single-stage detection process, allowed YOLO to achieve impressive nearly real-time FPS results. Redmon et al. reported YOLO to sustain the average FPS of 45, while the most accurate version of the Faster-RCNN network had 7 FPS [15]. However, the proposed network still has multiple limitations, which were addressed in later iterations of the paper [57, 58].

The architecture consists of 24 cascaded convolutional and 2 FC layers. The network is pre-trained on the ImageNet dataset [39]. The image size is additionally reduced by applying 1×1 convolutional layers in between, also referred to as "reduction layers" [15]. The simplified architecture is presented in Figure 19.

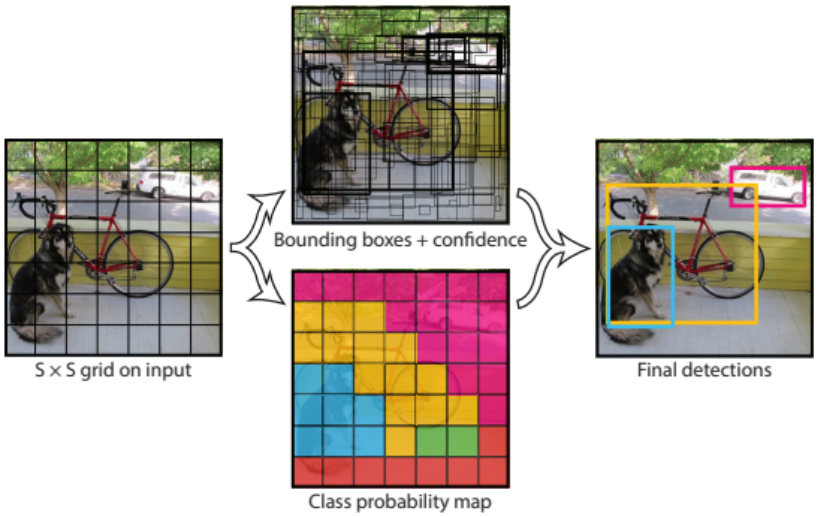


Figure 18: YOLO overview [15].

2.4.5 SSD

Another implementation of a single-stage network worth mentioning is a Single-Shot MultiBox Detector (SSD) [13]. The key differences in the architectures of SSD and YOLO are displayed in Figure 19. Liu et al. proposed a model that detects objects in real-time while preserving the accuracy. First, the image is fed into the backbone CNN, in the original experiments - VGG-16. The SSD head layers added after the backbone network are convolutional as well. Similarly to YOLO, the image is split into a grid of $n \times n$ size, where it benefits from the anchor boxes of varying aspect ratio.

However, as the objects might not always be within the grid boundaries, as can be noticed from Figure 20. Therefore, SSD paper introduced a concept of anchor boxes with offsets. The anchor boxes with offsets that intersect with the ground truth box of the object the most are then forwarded to the FC layers to predict the class and the location of the object.

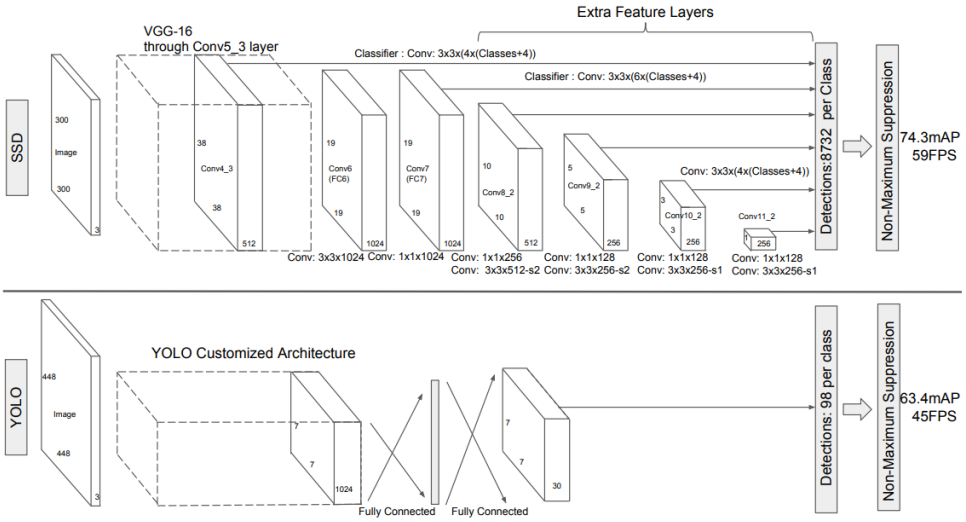


Figure 19: SSD compared to YOLO [13].

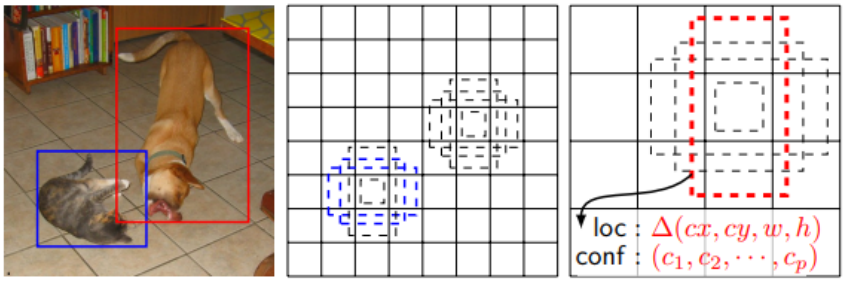


Figure 20: SSD anchor boxes [13].

The key object detection models of the last decade are presented in Table 1 below. As it can be deducted from Table, Faster-RCNN shows the best performance on PASCAL VOC, while YOLO has shown the highest FPS rate. The survey of Zaidi et al. [48] also discusses lightweight models. However, for the purpose of this study, the detection accuracy is the primary focus rather than speed, thus their review will be omitted.

As can be seen from Table 1, there are multiple models that are not covered in this section. Additionally, there are several detectors that show promising results

Table 1: Overview of object detectors [48].

Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	~0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	~0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	~0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	~3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	~4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
Swin-L	2021	HTC++	-	57.70%	-	-

[a]Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.Rows colored gray are real-time detectors (>30 FPS).

in the domain-adaptive object detection setup, such as FCOS [17] and DETR [16]. However, as their performance is not yet extensively evaluated, they will be omitted from this thesis.

2.5 Object detection in industrial environments

Although object detection is an extremely diverse field with applications varying from medicine [2] to self-driving vehicles [3, 4], the industrial object detection is mostly limited to safety and process monitoring [1, 5]. Fewer research has addressed the topic of industrial object detection. Normally, many of the introduced object detection methods can be applied to this problem directly without any significant modifications. Malburg et al. [7] has evaluated multiple different versions of object detection models such as YOLO [15] in an industrial setup. However, often in industrial environments severe challenges such as overlapping and occlusion take place. Moreover, the object instances often are represented in multiple scales. Therefore, it is often impossible to obtain reliable results using a standard object detector architecture. Wu et al. [6] has recently approached the problem of industrial object detection using an R-CNN based model that leverages Feature Pyramid Networks to improve detection at multiple scales and NMS with penalty factors to eliminate the occlusion problem. However, the proposed method does not consider the data availability limitations and confidentiality issues that were outlined in the Thesis objective. As an alternative, Kim et al. [8] attempted to minimize the need for labeling from a human by utilizing a deep active learning model. While the core of this approach is a standard Faster-RCNN [12] network, Kim et al. also proposed to utilize uncertainty evaluation in order to enable the model to label images interactively over time. As the training advances, the model evaluates whether the predicted bounding boxes are reliable or not using the sum of entropy for each bounding box. In the subsequent stage the

model requests the user to manually label 10% of the data with highest uncertainty. The model is then retrained with the updated knowledge. Although this approach seems promising, it does not consider the domain shift phenomenon that takes place in the setup defined by the Thesis objective. Later sections of the thesis will address this phenomenon.

2.6 Transfer learning

In this section, multiple transfer learning(TL) techniques will be introduced. Despite the fact that the methods discussed in the previous chapters have been successfully implemented in various scenarios, ML in general often struggles to apply such methods in real life. This is normally caused by insufficient training data, as it is often expensive to collect it, both in terms of time, financially, and sometimes simply not possible at all due to data accessibility issues [59]. Moreover, the requirement of having to train the models on new massive datasets often make ML solutions inefficient in practice. For this reason, TL concepts have been found advantageous as they address transferring knowledge gained from one task, domain or distribution to another. Here and in the subsequent sections, the notations are identical to those outlined by Pan et al. [60] and are introduced as follows:

- A domain $\mathcal{D}$ can be defined as a composite term, which is characterized by two elements: a feature space $\mathcal{X}$ and its marginal probability distribution $P(X)$; $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. With this being said, two domains are defined as distinct if they have either a different feature space or a different marginal probability distributions. In this thesis, the number of domains is restricted with one source and one target domains $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$, respectively [60].

- Equivalently, a task $\mathcal{T}$ can be characterized by its label space $\mathcal{Y}$ and the conditional probability of $\mathcal{Y}$ given X, e.g. $P(Y \mid X)$, $Y = \{y_1, \ldots, y_n\} \in \mathcal{Y}$. In practice, a conditional probability is defined as a function that can learn to predict a label y_i given a sample vector x_i [60].

Sun et al. [61] describes TL as a method to transfer knowledge when two given source and target domains are different, meaning that their feature spaces are separate e.g. $\mathcal{X}_S \neq \mathcal{X}_T$. Pan et al. [60] mentions that transfer learning can be also defined when the marginal probability is different, meaning that $P_S(X) \neq P_T(X)$. Similarly, transfer learning can be applied when $\mathcal{T}_S \neq \mathcal{T}_T$. An example of a simple TL problem is training a model that was trained for classifying cats, but instead is required to learn a new task such as classifying dogs. However, in various scenarios, the task is essentially the same but the domain is similar yet different. For instance, a model that was trained to identify cats in sketches, is instead required to identify cats in real images. Solving such tasks is the main focus of domain adaptation(DA). Often the terms "domain adaptation" and "transfer learning" are used interchangeably. However, according to Wang et al. [62] and Zhang et al. [63], DA is a special case of TL. In scientific terms, DA aims to align two domains when the source feature space matches the target feature space i.e. $\mathcal{X}_S = \mathcal{X}_T$, but the marginal probability

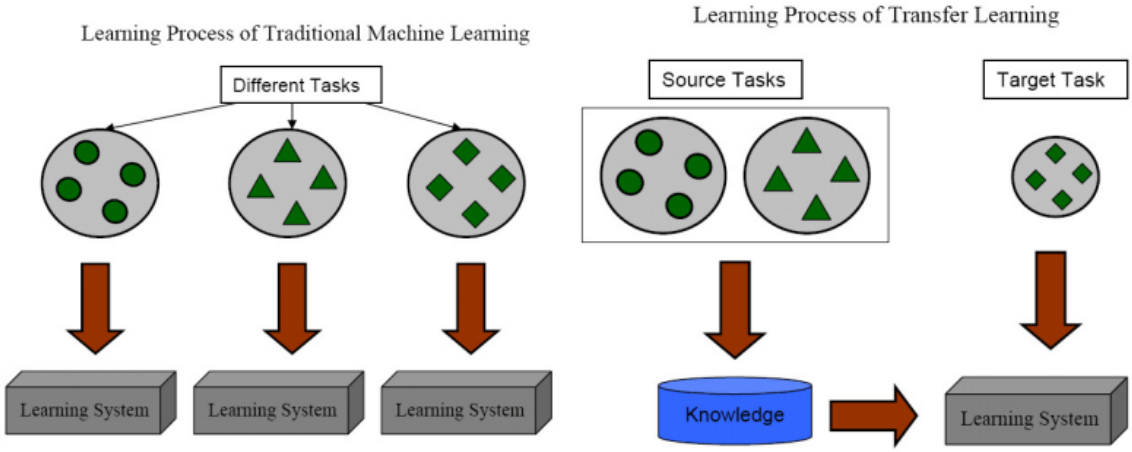


Figure 21: Comparison of ML to TL [60].

distribution is different: $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$ [61]. This is not to be confused with semi-supervised ML, where the source of both labeled and unlabeled data is typically originating from the same domain [61].

2.6.1 Domain adaptation

Lately, DA has been gaining popularity in both image classification and object detection tasks [63]. Typically, DA is used to predict a label given the source domain data and limited to none from the target domain. Most importantly, DA addresses the domain shift problem [63]. In a DA problem, a domain shift, also known as a distributional shift or dataset bias, can be defined as a change in distribution of data between source and target domains.

Zhang et al. [63] classify DA methods into three categories that are similar to ML types - supervised, semi-supervised, and unsupervised DA. Alternatively, Oza et al. [64] classify the collected DA methods into semi-supervised, weakly-supervised and unsupervised DA. Indeed, supervised DA is not commonly used since the primary goal of DA is to reduce the domain gap when the availability of data is limited. According to Zhang et al. [63], different methods attempt to solve the distribution shift problem by either minimizing the distance between marginal, conditional or joint distributions. The visual representation of these distribution alignment types are shown in Figure 22 and the methods in the following subsections attempt to minimize the domain shift by one way or another.

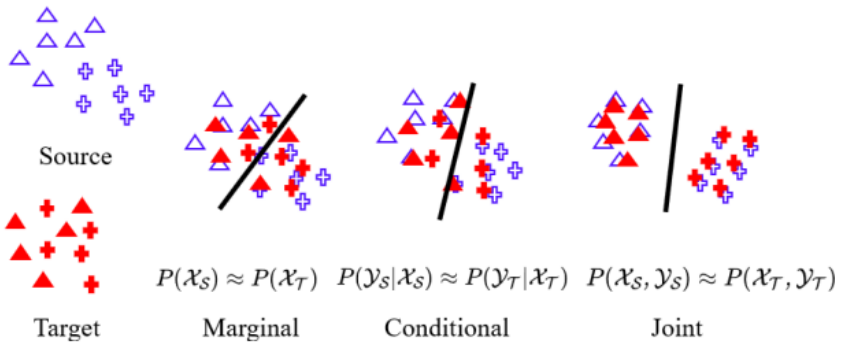


Figure 22: Distribution alignment types [63].

2.7 Domain-adaptive object detection

Next subsection will discuss some of the methods that were proposed at different times to solve the problem of domain shift in object detectors. Oza et al. [64] have extensively reviewed and grouped the existing approaches into following six categories:

1. Adversarial feature learning
2. Pseudo-label based self-training
3. Image-to-image translation
4. Domain randomization
5. Mean-teacher training
6. Graph reasoning

Many of the methods collected by Oza et al. overlap with each other and fall into more than one group. In this thesis, a few methods will be briefly reviewed for each of the categories above. Some of the methods are listed in Figure 23.

In the following methods, it will be assumed that $\mathcal{D}_S$ and $\mathcal{D}_T$ originate from similar yet different distributions. Additionally, the papers introduced in the following section address unsupervised domain adaptation(UDA) problem, which is naturally more sophisticated than a supervised DA.

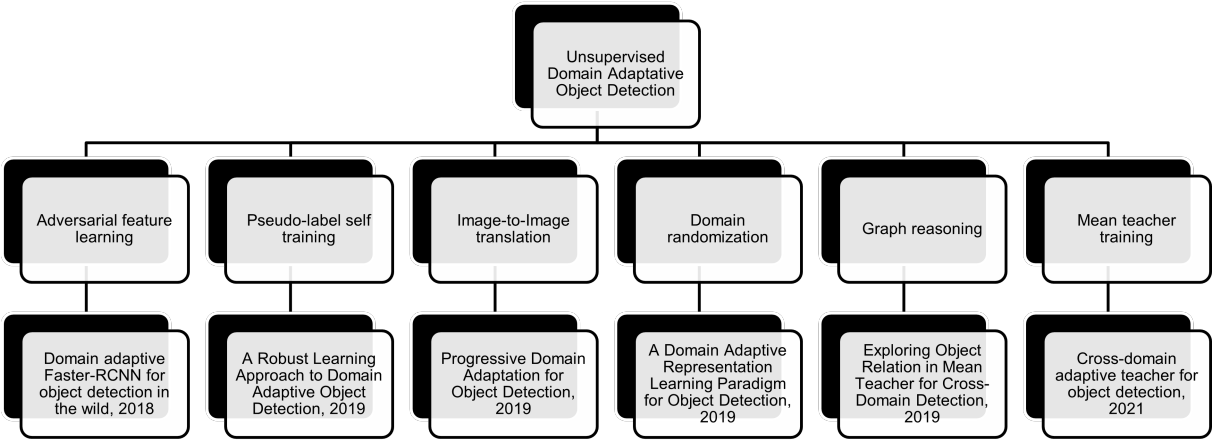


Figure 23: Unsupervised Domain Adaptive Object Detection.

2.7.1 Gradient reversal layer

One of the key components in a typical domain adaptive setup for both image classification and object detection problems is a gradient reversal layer (GRL) that has been introduced by Ganin et al. [18]. The authors suggest that in order to successfully align two domains, the proposed features should be as indistinguishable between the source and target domains as possible. In other words, the network should propose features that are familiar both domains. Figure 24 illustrates the Domain-Adversarial Neural Network (DANN), which implements such logic [18].

DANN is essentially a simple classifier network with a feature extractor and a predictor that returns a label for the input image. Two additional components are attached to the output of the feature extractor - a GRL and a domain classifier. On the forward pass of the DANN, the network attempts to predict the class and the domain labels and GRL behaves as a standard identity function $\mathbf{I}$. During the backpropogation, the GRL multiplies the gradient of the domain classifier by a fixed negative weight constant λ. This enables the domain classier to magnify the loss of domain classification and, therefore, "confuse" the feature extractor and force it to generate only domain invariant features [18]. Assuming that the GRL can be described as a pseudo-function $R_\lambda(\mathbf{x})$, where $\mathbf{x}$ is a feature vector at the input, the logic of a GRL is described in the equations below:

$$\begin{aligned} R_\lambda(\mathbf{x}) &= \mathbf{x} \\ \frac{dR_\lambda}{d\mathbf{x}} &= -\lambda\mathbf{I}, \end{aligned} \tag{4}$$

where the equations denote the forward- and backpropagation components, respectively.

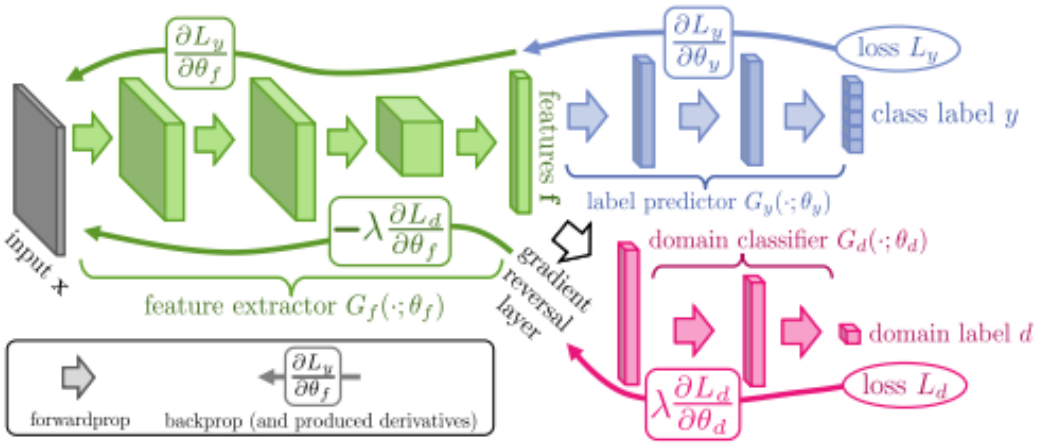


Figure 24: Domain-adversarial neural network and GRL [18].

2.7.2 Adversarial feature learning

Although DANN implementation is originally meant for domain adaptive image classification, GRL is a fundamental component in adversarial feature learning of object detectors and will be referred to in the subsequent sections of the thesis. An adversarial learning approach is typically class-agnostic [67], meaning that it aims to align the domains rather than the classes in the dataset.

The majority of the methods, collected in the survey by Oza et al. [64], are based on the two-stage object detectors and Faster-RCNN [12] in particular. This has been presumably facilitated by Pytorch [65] package in Python and frameworks such as Detectron [66] and Detectron2 [21] that enabled researchers to extend the possibilities of Faster-RCNN and improve its scalability. With the arrival of the mentioned tools and their pre-trained models, Faster-RCNN became a good starting point to experiment with new tasks, which was done by replacing backbone networks and adding new components.

One of the first implementations of such approach in object detectors is presented in the paper by Chen et al. [22]. The suggested architecture is based on the Faster-RCNN object detector. As it can be seen from Figure 25, this method proposed to apply adversarial learning at multiple stages of the detection by applying the GRL strategy, namely in the image- and instance-levels of the network. To be more precise, a GRL and a domain classifier are appended to the extracted feature map, same way as in Figure 24 to form an image-level domain classifier. Similarly, regions predicted by the FC layers of the Faster-RCNN network follow an equivalent procedure. Finally, the classifiers are regularized using consistency loss.

Here, to maximize the domain classification losses L_{inst} and L_{img}, they are calculated as shown in Equation 5.

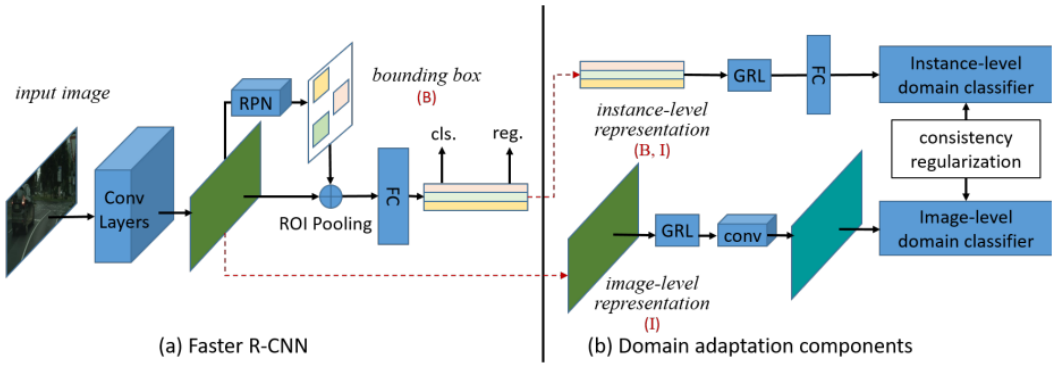


Figure 25: Domain Adaptive Faster R-CNN for Object Detection in the Wild [22].

$$\begin{aligned} \mathcal{L}_{img} &= -\sum_{i,u,v} \left[D_i \log p_i^{(u,v)} + (1 - D_i) \log \left(1 - p_i^{(u,v)}\right) \right] \\ \mathcal{L}_{ins} &= -\sum_{i,j} \left[D_i \log p_{i,j} + (1 - D_i) \log \left(1 - p_{i,j}\right) \right] \end{aligned}, \tag{5}$$

where $D_i = 0$ is a ground-truth label of the sample image i that originates from the source domain and $D_i = 1$ originates from the target domain; $p_{i,j}$ is a prediction output of the instance-level classifier for j-th proposed region of the sample image i; $p_i^{(u,v)}$ is a prediction output of the image-level domain classifier for the feature map region (u, v) of a sample image i [22]. Finally, two domain classifiers are regularized using consistency loss $\lambda \mathcal{L}_{\text{consistency}}$ [22] as shown in Equation 6.

$$\mathcal{L}_{\text{consistency}} = \sum_{i,j} \left\| \frac{1}{|N|} \sum_{u,v} p_i^{(u,v)} - p_{i,j} \right\|_2, \tag{6}$$

Due to the fact that the image-level classifier produces multiple N predictions per image, first the average of them is calculated, and then the Euclidean (or L2 norm) distance is measured between the predictions of the image- and instance-level domain classifiers [22].

The final objective of the network is then defined by minimizing the detection loss, while maximizing the domain classification losses at the image-level $\mathcal{L}_{img}$ and the instance-level $\mathcal{L}_{inst}$. The network additionally aims to minimize the consistency loss $\lambda \mathcal{L}_{\text{consistency}}$ between two classifiers [22].

Another most recent study made by Rezaeianaran et al. [67] proposed a different approach that compares the adversarial training to contrastive learning. Similarly to Chen et al. [22], the network with adversarial learning leverages a Faster-RCNN detector with instance- and image-level alignments. Rezaeianaran et al. also attempted a different approach for instance-level alignment by pushing the features closer in both domains if they represent the same class, and push them apart otherwise. This

is performed by utilizing a max-margin contrastive loss. The margin here denotes how far the features can be in order to be considered to represent the same class. According to Rezaeianaran et al. [67], a general constrastive loss term takes the form of Equation 7:

$$\mathcal{L}_{CL} = \sum_{i}^{N} \left[\left\| \mathcal{F}_S^i - \mathcal{F}_T^i \right\|_2^2 + \sum_{j,j\neq i}^{N} \max \left\{ 0, m - \left\| \mathcal{F}_S^i - \mathcal{F}_T^j \right\|_2^2 \right\} \right], \tag{7}$$

where m is the max-margin value and N is the number of classes.

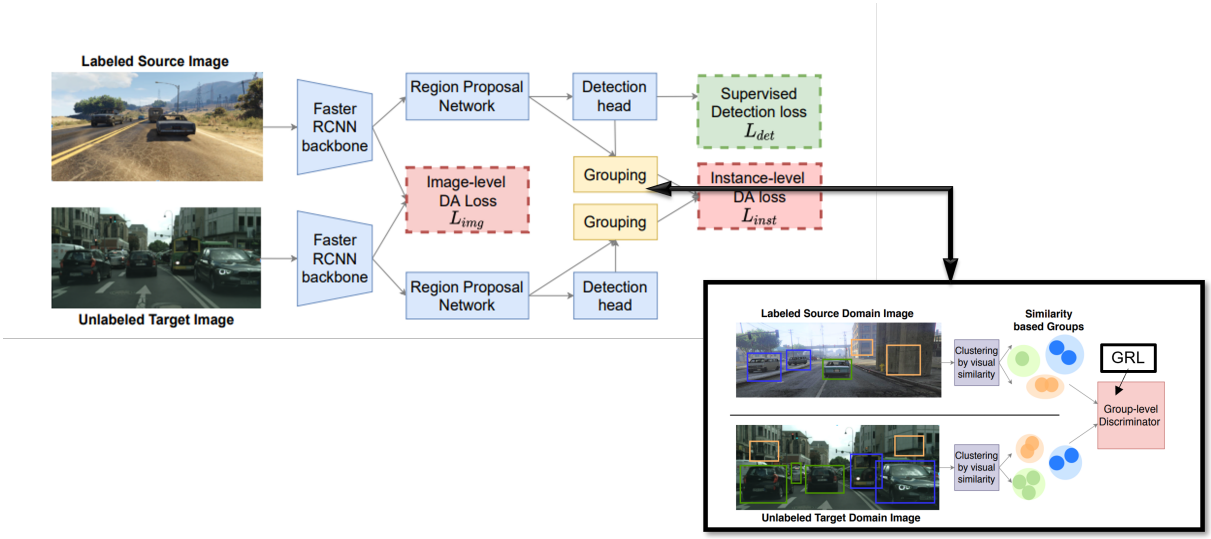


Figure 26: Seeking Similarities over Differences: Similarity-based Domain Alignment for Adaptive Object Detection, adapted from [67].

Rezaeianaran et al. suggested that using only image-level domain adaptation might force the alignment of the background features relevant to the source domain (images from the simulator). For this reason, the authors proposed to first group the generated features using similarity-based clustering. Initially, all proposed regions are considered to be separate clusters. Next, at every iteration, two closest clusters are merged based on the cosine distance. After this algorithm has converged, the average of the instances in each cluster is generated. The aggregated average proposal is then processed on a group-level in a standard adversarial manner using a GRL and a domain classifier.

2.7.3 Pseudo-labeling based methods

Another relatively straightforward method to solve an object detection problem can be done by using pseudo-labels. A naive approach to pseudo-labeling is to first train the source dataset $\mathcal{D}_S = \{\mathcal{X}_S^i, \mathcal{Y}_S^i\}_{i=1}^{\mathcal{N}_S}$ and then run inference to produce pseudo-labels

on the images from the target dataset $\mathcal{D}_{\mathcal{T}} = \left\{\mathcal{X}_{\mathcal{T}}^{j}\right\}_{j=1}^{\mathcal{N}_{\mathcal{T}}}$. The resulted labels will then form a new dataset $\dot{\mathcal{D}}_{\mathcal{T}} = \left\{\mathcal{X}_{\mathcal{T}}^{j}, \mathcal{Y}_{\mathcal{T}}^{j}\right\}_{j=1}^{\mathcal{N}_{\mathcal{T}}}$ [64]. However, the results obtained will naturally be noisy and of poor quality. Khodabandeh et al. proposed a three-phase training process illustrated in Figure 27 below.

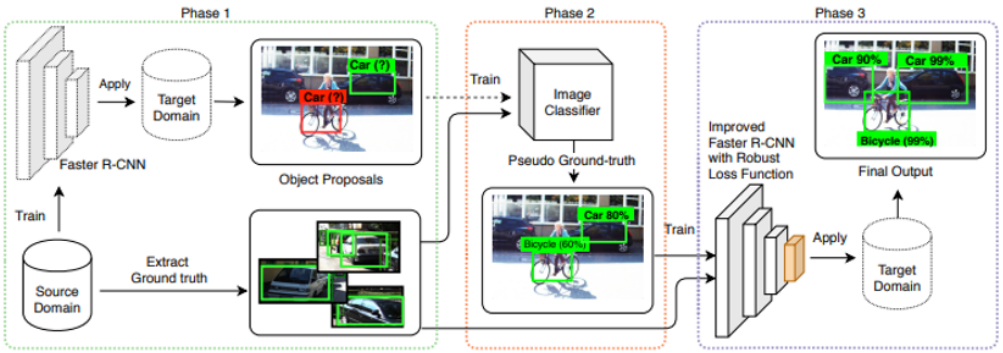


Figure 27: A Robust Learning Approach to Domain Adaptive Object Detection [68].

In the initial phase, the training is treated as if it was a regular Faster-RCNN network that is trained on the source dataset $\mathcal{D}_{\mathcal{S}}$. Next, the pseudo-labels are generated as explained earlier. In the following stage, the proposed regions are forwarded to the pre-trained image classifier, which allows to refine the model.

For the process of refinement, Khodabandeh et al. proposed to use the Kullback-Leibler divergence and defined the optimization objective for classification as follows:

$$\min_{q} \mathrm{KL}\left(q\left(y_c\right) \| p_{cls}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right) + \alpha\, \mathrm{KL}\left(q\left(y_c\right) || p_{img}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right), \tag{8}$$

where y_c is the class label, y_l is the bounding box location, α is a trade-off between two terms, p_{img} is the classification prediction of the image classification model and p_{cls} is the classification prediction of the Faster-RCNN detector model [68]. The goal of the refining process is to find a distribution $q(y_c)$ closest to p_{cls} and p_{img}. The process is relatively similar for the bounding box refinement and the reader is advised to refer to the original paper for more details [68].

The third stage of the process finalizes the strategy by retraining the final network with the labeled ground truth data from $\mathcal{D}_{\mathcal{S}}$ and the refined pseudo-labels from $\dot{\mathcal{D}}_{\mathcal{T}}$.

2.7.4 Image-to-Image translation

Another category that Oza et al. [64] outlined is image-to-image translation for UDA. Instead of trying to align features, this group of methods essentially attempt to pull the domains closer first. Hsu et al. [69] has suggested a Cycle-Generative Adversarial

Network(GAN)-based approach [70] to transform target domain images into source domain-alike images. Figure 28 represents the complete network introduced by Hsu et al.

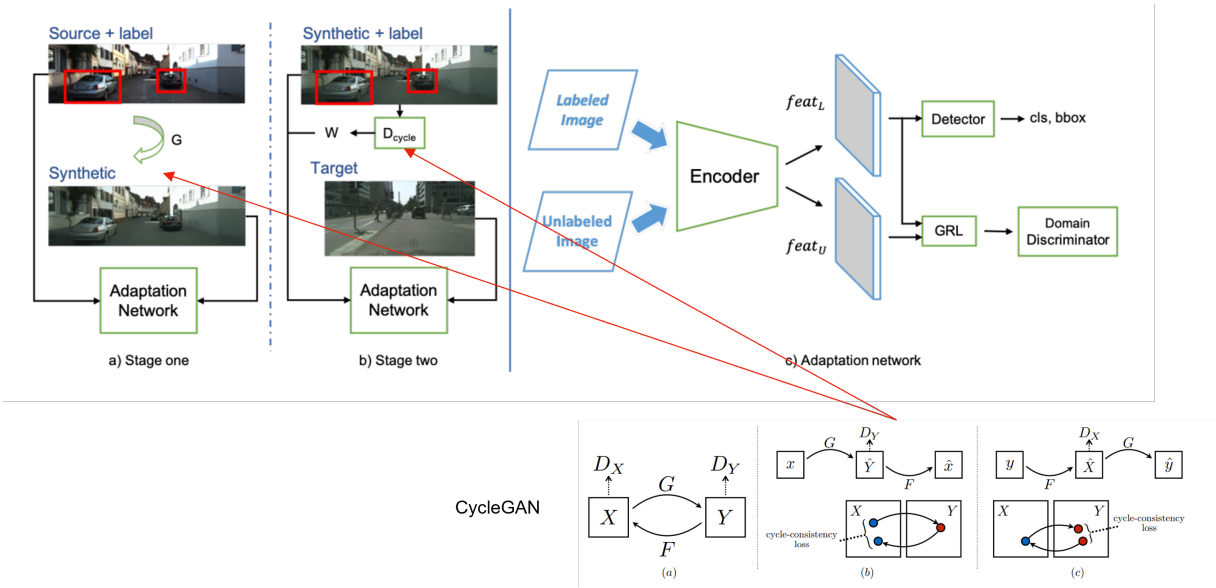


Figure 28: Progressive Domain Adaptation for Object Detection and CycleGAN, adapted from [69].

The generator part of the Cycle-GAN in Figure 28 (a) creates an intermediate domain from the source domain images. The source images are then forwarded to Faster-RCNN together with the labels. The network then tries to adapt the source domain to the synthetic domain. To further minimize the domain gap, adversarial learning techniques are used, such as a combination of a GRL and a domain classifier [69].

In Figure 28 (b), the synthetic source-alike images are then passed together with the inherited source labels back to the adaptation network to align new features with the target domain. In the second phase, Hsu et al. utilized the weights W from the discriminator of the Cycle-GAN, which was trained to be able to classify the source and the target domains. Ultimately, such approach allowed to amplify the importance of the synthetic samples that are more similar to the target domain. The detection loss was summed with the weighted loss from the discriminator and the adversarial loss to complete the adaptation from the synthetic to the target domain [69].

2.7.5 Domain randomization

Oza et al. [64] argues that often the accuracy of image-to-image translation methods is questionable as the domain shift between the synthetic and source domains still

exists. Slightly different approach has been offered by Kim et al. [71]. Instead of trying to pull two given domain distributions closer, they proposed a domain randomization technique, which generally attempts to generate a brand new domain that includes the same image in different style. This in practice allows the detector network to recognize features that are domain invariant and remove the domain bias.

Kim et al. introduced a detector network based on Faster-RCNN with two additional components. The first component is a domain diversification module. Similarly to Hsu et al. [69], the module in this implementation leverages a Cycle-GAN [70] for domain alignment. However, the diversification module is used to generate images in a fixed set of new domains rather than to match the source dataset with the target. The second component is a multi-domain discriminator, which is essentially an adversarial feature learning approach, but instead of trying to confuse the detector in a binary set of domains, it tries to learn domain invariant features of multiple additional domains. The architecture of such domain randomization method is presented in Figure 29.

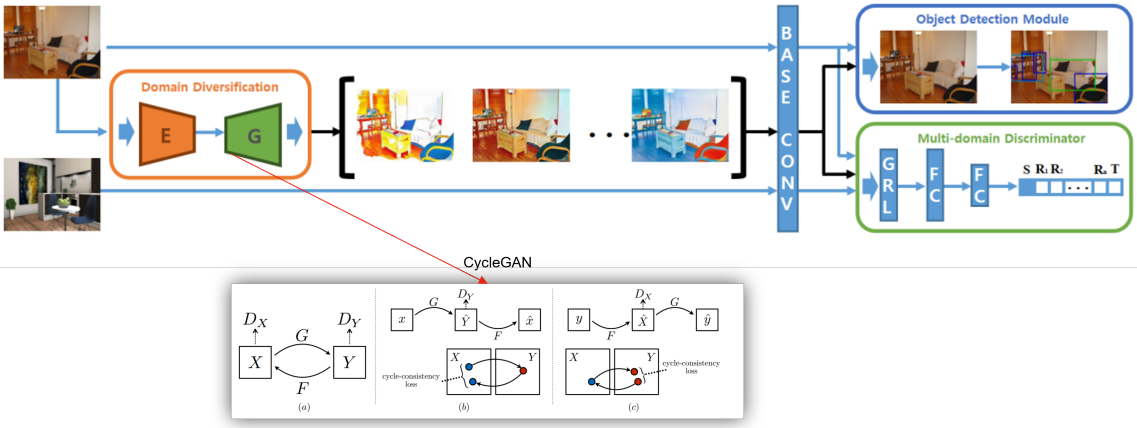


Figure 29: Diversify and Match: A Domain Adaptive Representation Learning Paradigm for Object Detection, adapted from [71].

During Cycle-GAN image generation, two additional loss terms were used to limit the randomization of an image - color preservation and reconstruction constraints. This was done in order to preserve features of the original image as it would otherwise start to negatively affect the model [64]. In the original experiments, Kim et al. considered three additional domains: a color preserved domain, a reconstructed domain, and a domain that combines both. Images from all the domains are then fed into the detector along with the inherited source labels to train a domain-invariant network with the help of a multi-class domain classifier and the resulted model was used to verify the performance on the target dataset.

2.7.6 Graph reasoning

Another common approach utilized in domain adaptation and transfer learning in general is mean teacher training. A typical mean teacher setup consists of two equivalent models. However, these models are trained using two separate strategies to adapt the detector network. On the other hand, graph reasoning based approaches of UDA have been gaining popularity not only in image classification problems, but also in object detectors. One potential reason for this is because graph models are easily applicable with other adaptation methodologies [64]. Cai et al. [72] proposed an architecture that combines both mean teacher and graph reasoning techniques in one solution and such architecture is presented in Figure 30.

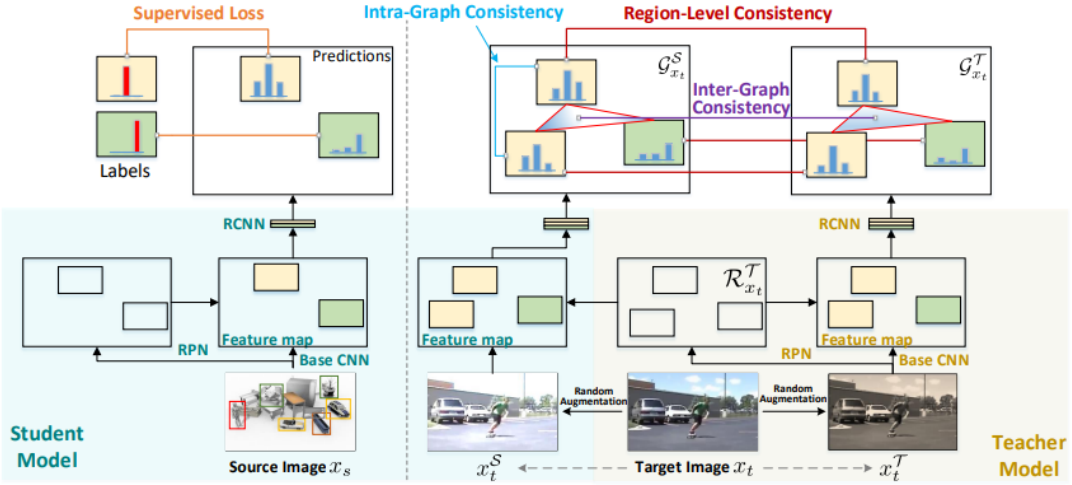


Figure 30: Exploring Object Relation in Mean Teacher for Cross-Domain Detection [72].

The term "graphs" represents a complex data structure that contains multiple nodes or vertices connected to each other via edges. In case of an image represented as a graph, each pixel can be considered as a node, which is linked to all of its neighbors. The setup developed by Cai et al. [72] introduces a student-teacher framework based on Faster-RCNN that verifies the similarity of the graphs at three different levels using regional-level consistency, intra-graph consistency and inter-graph consistency.

The training pipeline is split into two parts. Source domain images are trained in a locked student environment in a supervised fashion. Meanwhile, images from the target domain are augmented in two steps. First, images are randomly cropped, padded or flipped. These images are passed through the pre-trained supervised student model that generates predictions. At the same time, the original target images are augmented with color jittering or corrupted with noise. This set of images is then sent to the teacher model. The teacher model also produces predictions, which

are then compared with the predictions from the first set of augmented images by utilizing region-level consistency. Essentially, the region-level consistency is calculated as MSE of the region-level prediction error in both the student and the teacher.

Inter-graph level consistency is used to verify the quality of the two graphs that the teacher and the student models have produced. Cosine distance is calculated between graph representations of the proposed regions to evaluate the similarity.

Finally, intra-level consistency is then calculated to measure the quality of predictions within the same class of the student model. However, since target domain has no labels included in the UDA setup, the closest prediction argmin($labels$) is used to produce pseudo-labels. The intra-level consistency loss is then calculated for any two instances of the same class in one graph. For a detailed calculation of the loss terms the readers are referred to the original paper by Cai et al. [72].

2.7.7 Mean teacher training

A purely mean-teacher-based semi-supervised approach has been proposed by Liu et al. [73] As can be identified from the overview illustrated in Figure 31, it includes two sequential stages. The training pipeline is split into training of two similar Faster-RCNN-based detectors. During the initial burn-in stage, a regular supervised training is carried out. Next, the teacher model is supplied with weakly-augmented data. On the other hand, the student model uses strongly-augmented images for training. According to Liu et al., while stronger augmentations are needed in the teacher model to improve performance and to address over-fitting, weaker augmentations are more preferable in the teacher model to be able to generate pseudo-labels of a decent quality. Resulted pseudo-labels are in turn used in the student model.

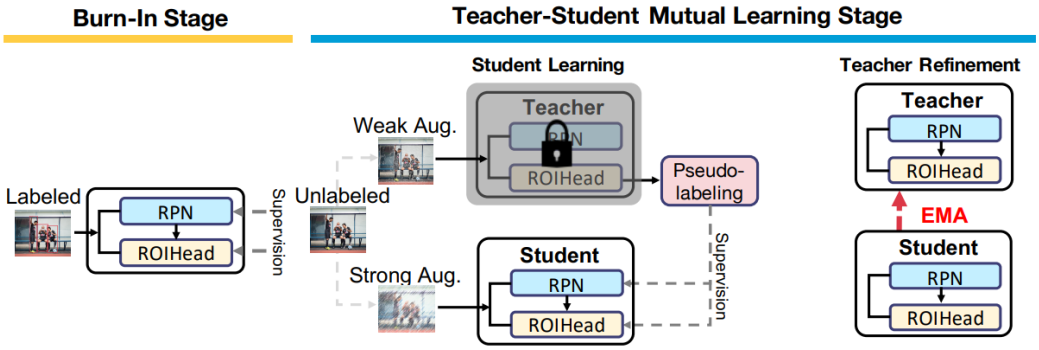


Figure 31: Unbiased Teacher for Semi-Supervised Object Detection [73].

In order to produce pseudo-labels of higher quality, Unbiased Teacher includes an Exponential Moving Average (EMA) module, along with a few other techniques [73]. EMA attempts to emphasize the most recent data by granting it a higher weight.

The EMA in Unbiased Teacher is defined in Equation 9 below:

$$\theta_t^i = \hat{\theta} - \gamma \sum_{j=1}^{i-1} \left(1 - \alpha^{-j+(i-1)}\right) \frac{\partial \left(\mathcal{L}_{\text{sup}} + \boldsymbol{\lambda}_u \mathcal{L}_{unsup}\right)}{\partial \theta_s^j}, \tag{9}$$

where $\hat{\theta}$ is the initial (burn-in stage) model weight, θ_t^i is the weight of the teacher model, θ_s^j is the weight of the student model at i-th and j-th iterations respectively. α is the trade-off weight for the EMA component in the training process and γ is the LR of the ensembled model [73].

Liu et al. also discusses the class imbalance problem that often causes the detector to learn underrepresented classes poorly [73]. In order to solve this problem, Liu et al. attempted to make use of the multi-focal loss [50], which attempts to put more weight on the samples with lower confidence, unlike a generic cross-entropy loss that treats all samples equally.

However, this method only solves a semi-supervised object detection without addressing the domain shift issue. Expanding the Unbiased Teacher method, Li et al. [20] revised the mean teacher training method to be applicable in a cross-domain adaptation setup. In addition to the original ensembled network proposed by Liu et al. [73], Adaptive Teacher architecture employs adversarial feature learning techniques to align the target and the source domains. A typical domain adaptation network, which includes a GRL and a domain classifier, is appended to the backbone feature extractor of the student model. The complete architecture is displayed in Figure 32.

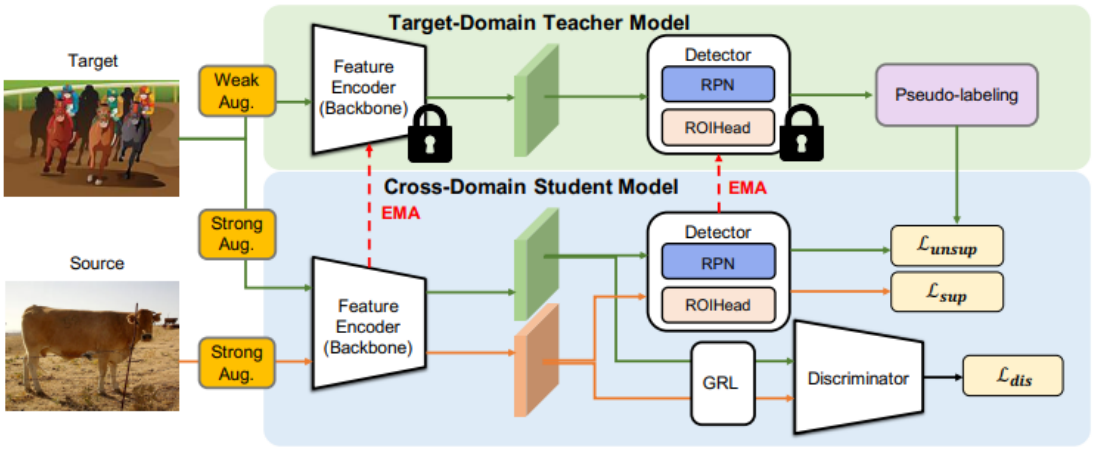


Figure 32: Cross-Domain Adaptive Teacher for Object Detection [20].

Liu et al. [73] specified that the teacher model is supplied with weakly-augmented target images, while the student model uses both strongly-augmented source and target images. Weak augmentations include cropping and flipping the image horizontally, and strong augmentations included grayscaling, color jittering, Gaussian

blurring and cutting out patches. The same strategy was employed in this Adaptive Teacher method. The augmentations used in these works [20, 73] are presented in Figure 33.

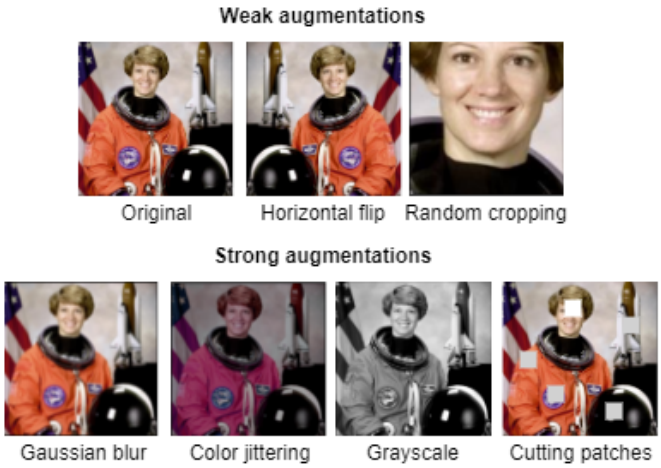


Figure 33: Augmentations used in Unbiased Teacher (adapted from the official Pytorch documentation [74]).

Although all of the methods discussed in this chapter are based on Faster-RCNN, which is a two-stage object detector, fewer papers also addressed a single-stage object detection in a domain adaptation setup, such as UDA for YOLO [75], [76], UDA for FCOS [77] and UDA for DETR [16], [78]. However, their review will be omitted and the grounds for it will be given in the Research Methodology section.

2.8 Continual learning

While traditional TL aims to apply knowledge collected from one domain to another, lifelong learning offers adaptive algorithms, which would accept a continuous stream of information that becomes available over time [23]. In case of the simple object detection problem, continual learning can be applied to a new task, such as to learn new classes of objects that are supplied progressively over time. This approach is potentially useful due to the scalability benefits it brings, since retraining the entire model every time a new object arrives to the database is computationally expensive.

Similarly to the human brain, ANNs in object detectors tend to forget old knowledge learned as the memory gets overwritten with fresh data. This phenomenon in continual learning is commonly addressed as "catastrophic forgetting" [23]. Parisi et al. summarizes the up-to-date methods of continual learning that are effective

against catastrophic forgetting into three categories: retraining with regularization, selective training with network expansion and retraining selective network with expansion. These methods are illustrated in Figure 34.

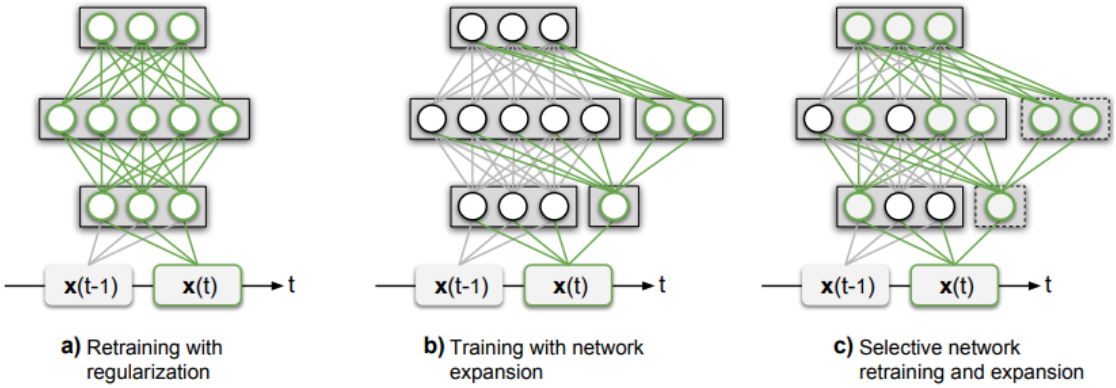


Figure 34: Continual learning approaches [23].

The methods proposed in Figure 34 (a) attempt to solve catastrophic forgetting by means of regularization. In these group of methods, the algorithm seeks to penalize the modifications in the model of the originally trained task. A simplistic approach of such method was presented by Razavian et al. [79], where the gradients calculation for the parameters of the original is disabled completely. Slightly adjusted version of this method was also proposed by Donahue et al. [80], where the LR was decreased to minimize the parameter update instead of blocking the update altogether.

In simplistic terms, the method of training with expansion denotes retraning the original network with an additional number of layers appended. A typical example of such a method was suggested by Rusu et al. [81], where the network was first trained on the initial task $\mathcal{T}_0$. As N new tasks were added, N number of layers were appended to the network, as shown in Figure 34 (b). The parameters of the original network were left unchanged, and only the additional connections were trained. Although the disadvantage of this method was that the network complexity would grow linearly over time as new tasks are added, the experiments delivered sufficient performance.

The method proposed by Yoon et al. [82] falls into the third category, which is illustrated in Figure 34 (c). The concept is fundamentally similar to the one suggested by Rusu et al. [81], with the exception that their model selectively retrains the network from the original task. Connecting the neurons sparsely reduced the computational overhead, as well as allowed the network to retain the previously learned tasks. Parisi et al. [23] have collected many other promising continual learning methods and readers are advised to refer to the original survey for more information.

In the context of the notation defined earlier in the Transfer learning subsection, continual learning can be expressed as training on the task $\mathcal{T}_n$ given the task $\mathcal{T}_{n-1}$ within the same domain or a set of domains $\mathcal{D}_n = \mathcal{D}_{n-1}$.

3 Research Methodology

In this section, an in-depth study will be conducted to solve the challenges outlined in the Thesis objective. The section will first discuss the process of dataset selection. Next, several object detection candidates will be analyzed. The selected object detection method will be examined in multiple domain-adaptive setups. Upon analyzing the existing DA methods, one candidate will be studied in greater detail to propose a novel architecture. Finally, continual learning will be investigated in the selected cross-domain object detection model.

3.1 Dataset

One of the objectives of this thesis was to implement an identification system using images of the equipment and images of the 3D models to expand the dataset. 3D images of the equipment can potentially be the main source of the dataset. Autodesk Navisworks offers a possibility of rendering both images and videos of the desired equipment. In order to make the process of data collection scalable, it was proposed to use Navisworks API to automate the process. Following the documentation of the API [83], a simple script with an extra toolbox was used to filter out the required model from Navisworks, to render and to export images in `.jpg` format. The results are presented in Figure 35 below.

Figure 35: Example of the rendered image of an arbitrary model.

It is possible to automatically rotate around the selected object in order to render images from all sides of the equipment as it is important to have variety while accumulating a dataset. However, due to the time constraints, this study will leverage existing open-source datasets to address the Thesis objective.

Previously, in the Neural networks in computer vision section, few datasets, such as ImageNet [39], COCO [41] and PASCAL VOC [40] were briefly mentioned. These datasets are universally used in image classification and object detection problems. However, in order to leave room for further research as well as to align with the objectives of the thesis, the dataset should ideally consist of industrial equipment and include corresponding 3D models for each object. Datasets such as ImageNet, PASCAL VOC and COCO typically contain generic objects that people face in everyday life.

Ultimately, it was concluded that the dataset named Texture-LESS (T-LESS) [19] meets the requirements. It consists of nearly 50 000 training and 10 000 testing images

of thirty industry-relevant objects. The training subset consists of rendered images in a simulated environment, while the test subset is taken in real-life conditions. Different objects in the dataset are often distantly similar to each other, which makes the task slightly more challenging. Finally, the dataset also includes Computer-Aided-Design(CAD) `.ply` files with 3D models of the objects, which can then be easily converted into any other required CAD format. The dataset is originally meant for 6D-pose estimation [19] as a part of the Benchmark for Pose Estimation(BOP) challenge [84]. As a result, the format of the dataset is derived from the format defined by BOP [85]. A script was developed to convert it into a format commonly used by the modern object detection frameworks, as well as to remove redundant information that is only applicable to pose estimation tasks. Initially, the dataset was converted to mimic YOLO [15] format, but later it was switched to PASCAL VOC due to its better flexibility in two-stage object detectors. An example of the annotated T-LESS image from a real environment is illustrated in Figure 36.

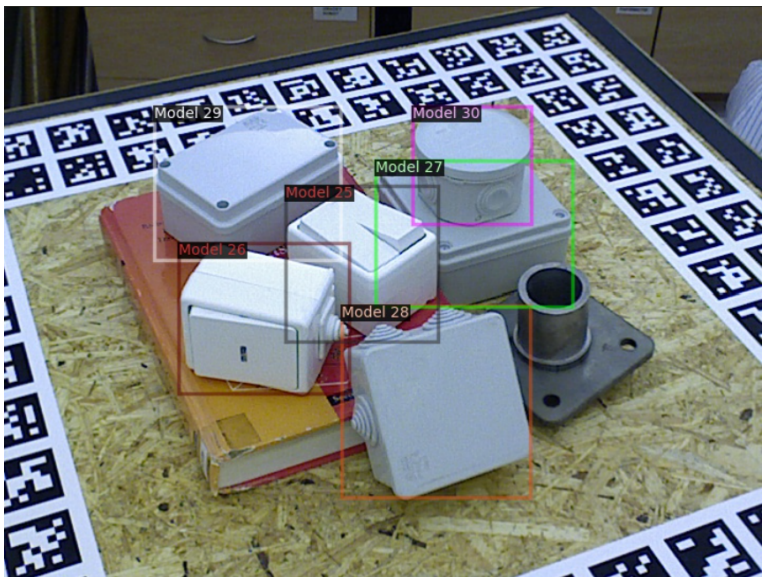


Figure 36: T-LESS real setup, labeled [19].

The names of the object classes presented in T-LESS, although undisclosed, are irrelevant to this thesis. Therefore, here and in the subsequent sections, the objects will be given arbitrary names in the format of "`Model` N", where N is the ID of the object. Additionally, in the experiments, the datasets with images from the simulator will be mainly referred to as "source" and the images from the real environment setup will be referred to as "target" datasets. The images from the source dataset are saved in `.jpg` format, while the real images are stored as `.png` files, identically as in the original T-LESS [19] dataset. Figure 37 illustrates the distribution of the instances by classes in the source dataset.

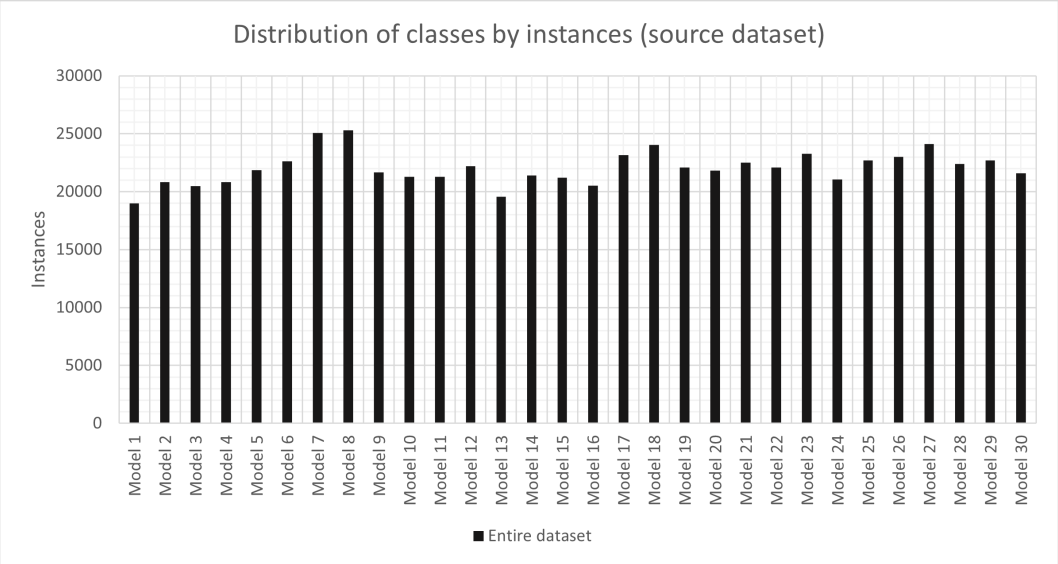


Figure 37: Distribution of the classes in the rendered subset of T-LESS dataset. Total number of images: 42 500. Total number of object instances: 661598.

As can be derived from the Thesis objective, the detection accuracy on the source dataset is irrelevant in this thesis. The primary goal is to evaluate the performance for the real images. Therefore, for this work, the source dataset is not divided into the training and testing subsets. On the other hand, the target dataset will be split in 85%/15% proportion. The distribution of classes by instances in the target dataset is presented in Figure 38 for both splits.

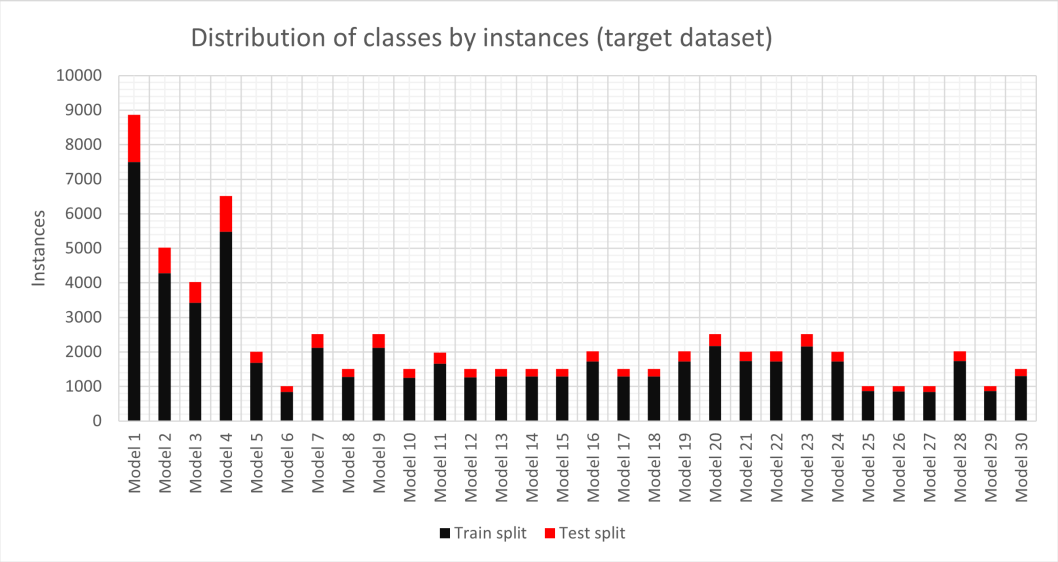


Figure 38: Distribution of the classes in the real subset of T-LESS dataset. 8 568 and 1 512 (85/15 %) training and testing images, respectively. Total number of object instances: 58845 training and 10362 validation instances.

3.2 Preliminary experiments

In the long term, the idea behind the project is to implement a scalable system that would identify real-life images of the equipment given the 3D images from the simulator. Therefore, as a default setup it was decided to use the rendered T-LESS-based dataset for training, and the real T-LESS-based dataset for evaluation. Naturally, in object detection problems, choosing the right detector is just as important as preparing the dataset. For the initial experiments, Faster-RCNN [52] model was tested.

Additionally, YOLOv3 [58] was considered. However, unlike Faster-RCNN, YOLO has not been as popular, according to Figure 17, especially in cross-domain adaptation problems. This was partially due to flexibility of Faster-RCNN when it comes to replacing different components of the network in order to improve the results. Although YOLO proved to be significantly faster than Faster-RCNN as discussed earlier in the YOLO section, in this thesis, the flexibility of the network and its accuracy are treated as higher priority tasks. For these reasons, Faster-RCNN will be used in all further experiments.

3.2.1 Metrics

As for the detection evaluation, here and in the further experiments and in order to match the selected dataset format, the metrics were employed in accordance with the mean average precision (mAP) metric of the PASCAL VOC [40] challenge. In order

to measure the mAP, first the confusion matrix should be constructed. Confusion matrix is a generic performance measurement tool in statistics and ML. According to PASCAL VOC, precision and recall are the key terms in object detection tasks that can be extracted from the confusion matrix. The equations for precision and recall can be derived from Table 2.

Table 2: Definition of confusion matrix and some of its terms.

		Predicted class		
		Positive	**Negative**	
Actual class	**Positive**	True Positive	False Negative	**Recall** $\frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$
	Negative	False Positive	True Negative	
		Precision $\frac{TP}{TP+FP} = \frac{TP}{all\ detections}$		**Accuracy** $\frac{TP+TN}{TP+TN+FP+FN}$

To calculate precision and recall, it is important to define how the True Positive (TP) term is calculated. Unlike, in image classification tasks, this calculation is not as straightforward. Since the objects need to be localized correctly as well as identified, comparing ground-truth labels against the predicted classes is not enough. This issue was addressed in COCO challenge [41] by introducing the Intersection-over-Union (IoU) metric. The principles of precision, recall and IoU in object detection are presented in Figure 39. As the model proposes an anchor box, the classifier outputs a confidence score, which is a probability of the box to contain an object.

The region is then additionally compared against the ground truth annotations of the bounding boxes to calculate the IoU value, also known as the Jaccard distance. Finally, if IoU is above the pre-defined threshold, the predicted object is considered to be a TP. In PASCAL VOC evaluation, the default threshold is 0.5 [86] and only predictions with the highest confidence scores are counted.

The prediction is labeled as False Positive (FP) when either the predicted class is wrong or its IoU is below the threshold. Otherwise, if the confidence score of the proposed region is below the threshold, the value is labeled as False Negative (FN). Next, precision and recall values are calculated as specified in Table 2.

Another key term in object detection metrics is Average Precision (AP). AP is commonly known as the area under the Precision-to-Recall curve. Pairs of values for precision and recall are recorded at multiple intervals and added to the Precision-to-Recall curve. In PASCAL VOC 2010-2012 [40], the area under the curve is interpolated first, as shown in Figure 40. The interpolation p_{interp} of the curve p is performed using Equation 10, where r is a recall level.

$$p_{\text{interp}}(r) = \max_{r' \geq r} p(r') \text{ [86]} \tag{10}$$

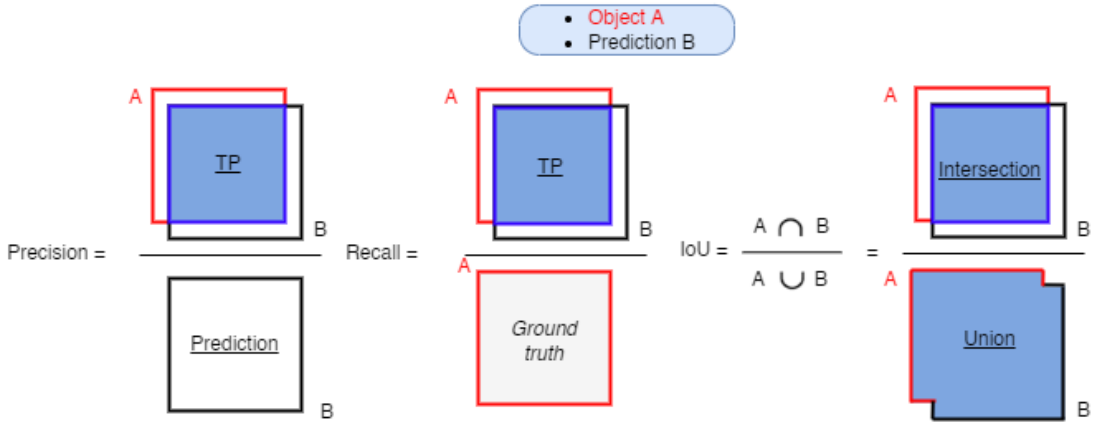


Figure 39: Visual representation of the terms used in the PASCAL VOC metrics.

Then, the integral value under the smoothed curve is calculated. The main objective of the AP metric is to maximize the area under this curve.

In the following sections, the terms mAP and `AP` will be used interchangeably. Generally speaking, `AP50` stands for the AP at IoU value of 0.5 and `AP75` stands for AP at IOU of 0.75, as defined by COCO [41]. With this in mind, all the experiments listed in the thesis will measure the `AP50` metric to evaluate the methods.

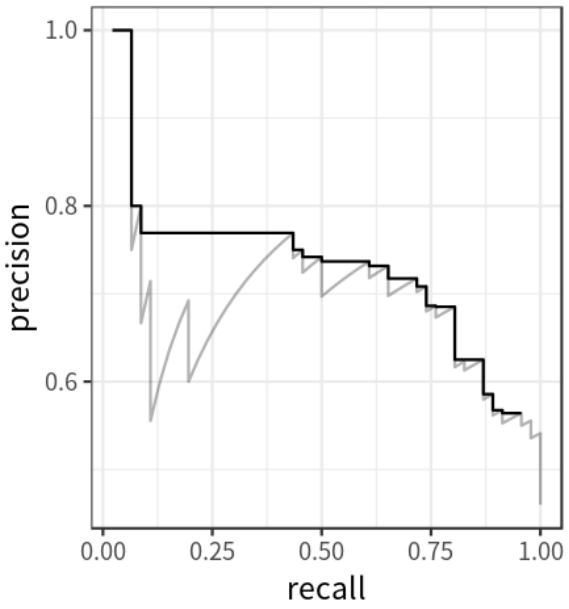


Figure 40: Smoothed average precision curve [86].

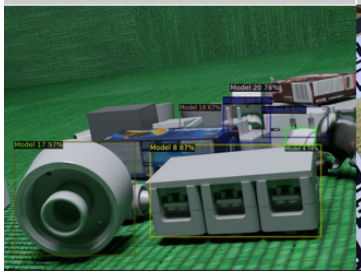
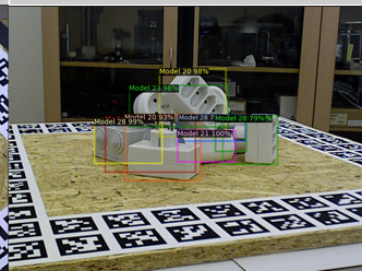
3.2.2 Naive object detection approach

Due to the simplicity of setup, Faster-RCNN [12] was selected as the first candidate for the object detection pipeline. In this setup, Faster-RCNN detector was configured to use in Detectron2 [21] and was evaluated on the dataset presented in the Dataset section. For the initial experiments, a Faster-RCNN network with a ResNet backbone was used as it was concluded to be the best among the networks listed in Section 2.3. The ResNet architecture had 50 layers. The training has been conducted in three stages. First, the model was trained on the source dataset. Next, this model was evaluated on the real images of the same classes. Finally, an identical model has been trained on the target dataset for comparison (oracle).

The evaluation in all three stages followed the metrics definition outlined in the Metrics section. After training the model for a short duration, corresponding to 20 000 iterations and with the learning rate `BASE_LR` = 0.00125, it quickly became evident that the solution has to be more complex in order to solve the challenges outlined in the Thesis objective. The results of the initial experiments are shown in Figure 3.

As it can be concluded, there is a significant performance drop when the environment is slightly changed. This is essentially a result of two related problems combined: over-fitting and domain shift. Naturally, the regular detector fails when two datasets originate from different environments and not only their background, but also their lighting conditions, positioning and textures deviate from one another. Moreover, because the target dataset is substantially smaller, there is a chance that

Table 3: Experiments with a simple Faster-RCNN model.

Model trained on rendered data and tested on rendered images	Model trained on rendered data and tested on real images	Model trained on real data and tested on real images
AP - 64.3510 AP50 - 76.0020 AP75 - 72.4191	AP - 9.0097 AP50 - 13.6304 AP75 - 10.046	AP - 83.3187 AP50 - 98.1990 AP75 - 94.2122
		

the model overfits in the third training stage. Nevertheless, it is believed that with right hyperparameters and a carefully refined training process with longer training time, the results can be improved, though not significantly.

3.2.3 Experiments with classical domain-adaptive methods

To overcome the domain shift phenomenon, it was proposed to experiment with the existing domain adaptation solutions. For these experiments, two existing open-source methods were suggested.

In the initial study, a model based on decoupled adaptation for cross-domain object detection [87] was tested. The model introduced by Jiang et al. essentially proposes an Adversarial feature learning approach, where a GRL is applied to the classifier and the box regressor in a decoupled way, i.e. the problem is split into two sub-problems to avoid them from interfering with each other. According to Jiang et al., this could improve the discriminability of the detector. Readers can refer to the original paper [87] for more information. The results of the decoupled adaptation experiments are outlined in Table 4.

First, a model was trained on the source dataset. Similarly to the Naive object detection approach, the performance drops dramatically when testing the model on the target dataset. However, after training the adaptation network, the results have improved by a considerable margin. The final result on the combined dataset is `AP50` = 64.32%. The result has significantly improved over the experiment without any adaptation presented in Table 3 with `AP50` = 13.63%.

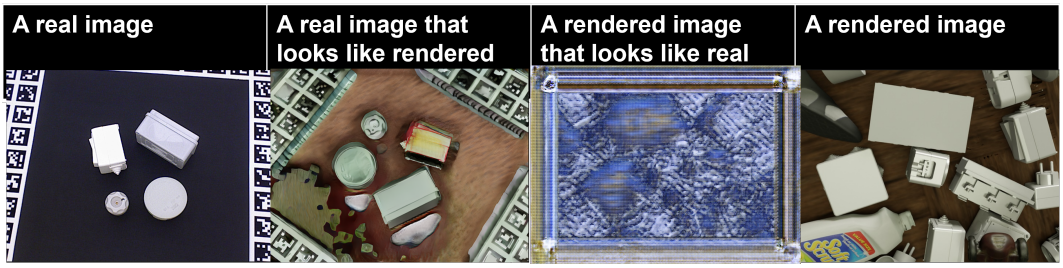
Recently, many open-source solutions proposed [88, 89, 90] an Image-to-Image

Table 4: Results of the experiments with a D-Adapt based method.

	Test on source data	Test on external (real data)
Model trained with only rendered data	AP 59.39 (AP50 - 74.10, AP75 – 67.93)	AP 6.77 (AP50 - 13.83, AP75 – 5.79)
Model trained on rendered data and adapted to real	AP 57.69 (AP50 - 73.11, AP75 – 67.01)	AP 36.85 (AP50 - 64.32, AP75 – 38.02)

translation approach in a cross-domain object detection setup. Following Zhu et al., Cycle-GAN [70] was used in an attempt to generate an intermediate domain. The results obtained after training the model for 44 epochs (= 36 hours) are compiled in Table 5.

Table 5: Results of the experiments with Cycle-GAN.



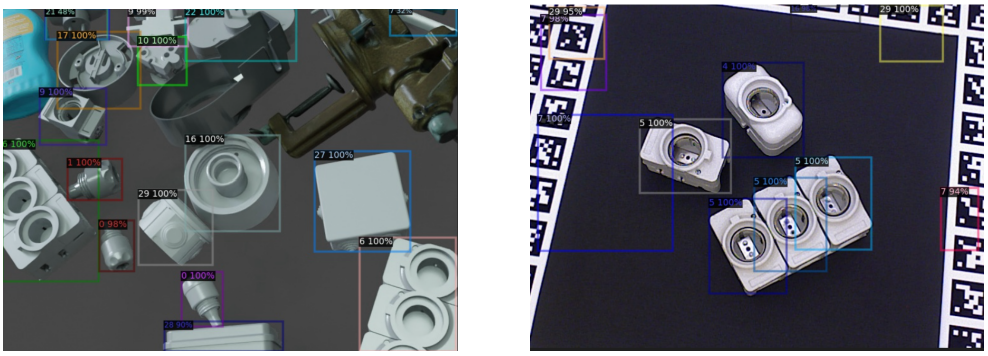
As it can be noted, even after an extensive training with a massive dataset of nearly 50 000 images, Cycle-GAN still produced fairly low-quality results, especially for the real-alike images. This problem has also been acknowledged by Zhu et al. as a limitation of Cycle-GAN [70], where differences in the distribution characteristics of the training dataset caused similar artifacts. Due to the low performance on the T-LESS dataset and long training time, this method was excluded from further tests.

3.3 Experiments with Adaptive Teacher

For the next set of experiments, this thesis proposes to analyze an ensembled setup. An ensembled setup is a combination of multiple algorithms that leverage their benefits to produce a superior result. In regards to domain-adaptive object detection, such setups were summarized earlier in the Mean teacher training section. As a base study for this thesis, the Adaptive Teacher [20] model is studied extensively. Unlike the architectures in the previous experiments, mean teacher training frameworks

implement multiple adaptation strategies. In case of the Adaptive Teacher method, Li et al. proposed to use both adversarial and pseudo-labeling techniques that are embedded into the student-teacher framework.

To validate the suitability of this architecture, the code base has been modified to utilize the prepared T-LESS dataset and was trained as it is. Figure 41 illustrates the results of the visualization after training such model. Similarly, to the previous experiments, the model was based on Faster-RCNN [12] and a ResNet [45] backbone network with 101 layers, which was pre-trained on the ImageNet [39] dataset. Although traditional ML and DL algorithms utilize the term epoch, which stands for one forward and one backward passes of the loop for the entire set of the training samples, in Detectron2 [21] the term "iterations" is used instead. Iterations denote a number of times to complete the forward and backward passes for a batch of training samples, where a batch is a small subset of data. From here onwards, iterations will be used to measure the training duration due to simplicity of the comparison process.



(a) Visualization on the source dataset (b) Visualization on the target dataset

Figure 41: Results of the experiments with Adaptive Teacher as it is.

Naturally, the model performed poorly on the custom dataset without tweaking any hyperparameters. Although the performance on the source dataset is more than satisfactory (Figure 41 (a)), random artifacts take place in the form of noisy bounding boxes, as can be identified in Figure 41 (b). For this round of experiments, the number of iterations was set as high as possible in order to explore the limits of the setup. Although the model was trained for 360 000 iterations, such huge values are often redundant as the selected dataset has extensive amount of images and, thus, less iterations can be used, as will be presented in Results.

To overcome the issue of noisy predictions, two steps have been carried out. First, the LR scheduler has been adjusted in the original Adaptive Teacher [20] implementation. In the original setup, the Adaptive Teacher algorithm used a warm-up two-stage multi-step scheduler. This scheduler initializes the LR in two stages. First gradually increases for a duration of warm-up `WARMUP_ITERS` = 1000 iterations, until it reaches the fixed `BASE_LR` = 0.001. Instead, this thesis suggests to use a multi-stage cosine-annealing scheduler without restarts, as suggested by Loshchilov et al. [91]. In these experiments, the learning rate first slowly ascends to the `BASE_LR` = 0.001. However, as the ceiling is reached, it starts to decay to zero by

the time of the last iteration `MAX_ITERS` is reached. This in theory allows to protect the scheduler from overshooting as it approaches the global minimum. Although Loshchilov et al. additionally proposes resetting the scheduler after it reaches the minimum, in this thesis such implementation is neglected for the time being. Both schedulers are compared against each other and are illustrated in Figure 42.

Another issue that potentially caused the noisy predictions on the real images is over-fitting. This might happen when, given the model complexity, the model is trained for an excessive amount of time. As a result, the model would only perform well on the training data. Therefore, to identify a suitable number of iterations required for training, an early stoppage algorithm was added. Instead of checking whether the loss keeps decreasing, the `AP50` metrics is tracked for the validation set at every `EVAL_PERIOD` = 1000 interval. The best `AP50` value is stored for the total of `PATIENCE` = 10 evaluations. If the value does not improve for longer than `EVAL_PERIOD` × `PATIENCE` iterations, the best model is saved and the training is terminated. Otherwise, the best `AP50` is updated. The results of such modification are presented later in the Results section.

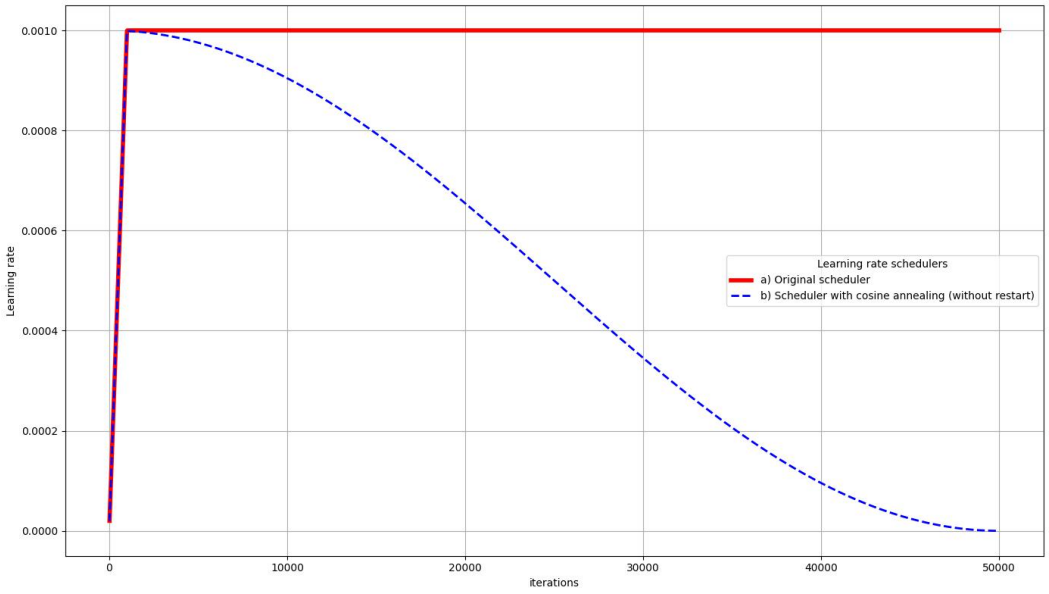


Figure 42: a) The original LR scheduler against b) The proposed cosine annealing LR scheduler without restarts.

3.4 Regularized Cross-Domain Adaptive Teacher

In the subsequent sections, a novel architecture will be introduced. The effectiveness of the existing methods has been evaluated to address domain shift between various open-source datasets. Most often DA is performed to transfer knowledge from one

dataset for autonomous driving (Cityscapes) to another (Foggy-Cityscapes, Sim10k, KITTI), or to transfer knowledge from one dataset with general objects (PASCAL VOC) to another (Clipart, Watercolor, Clipart) [64].

However, no comprehensive work have addressed the industrial cross-domain object detection. Additionally, motivated by works of Li et al. [20] and Xu et al. [92], this thesis assesses the practicality of the categorical regularization in the Adaptive Teacher framework. Unlike the original architecture of Adaptive Teacher [20], the model is aligned at two stages: image-level and instance-level. Following Chen et al. [22] and Xu et al. [92], a consistency loss term has been added to regularize both stages. The training process follows similar principles to the ones outlined in the Experiments with Adaptive Teacher section. The proposed architecture is presented in Figure 43.

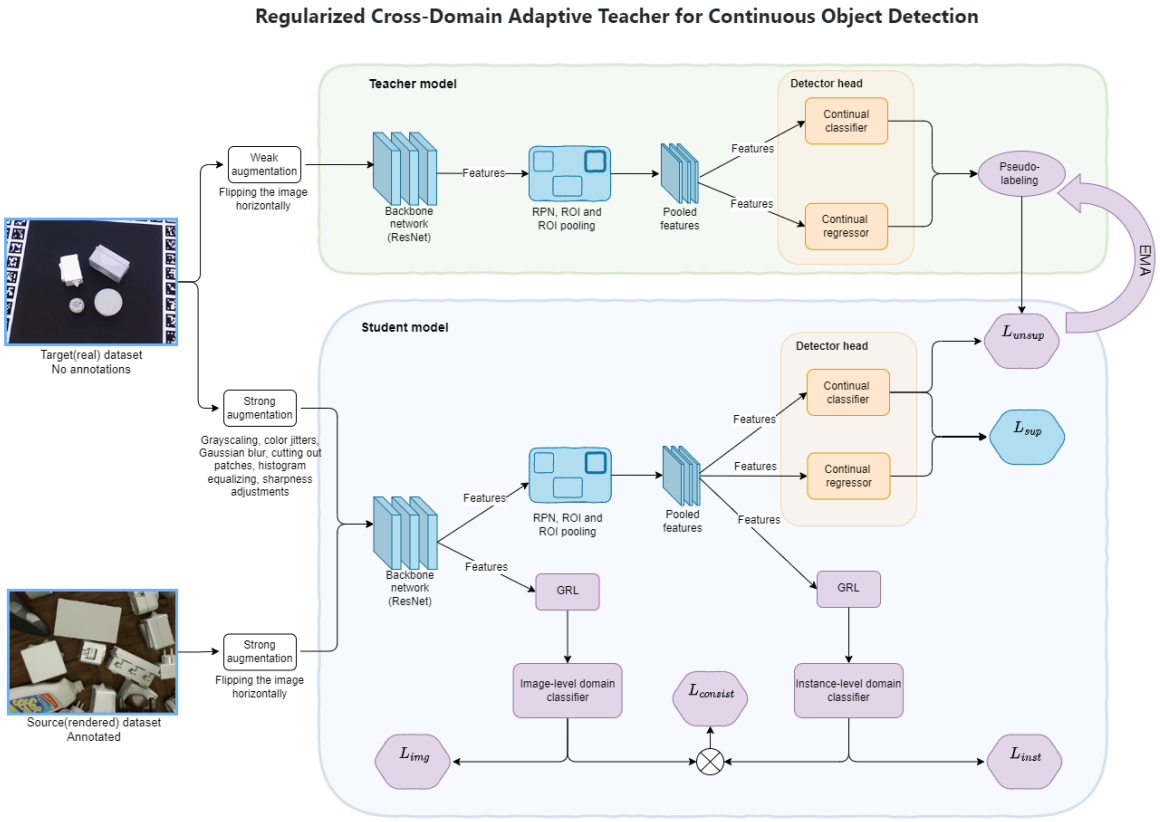


Figure 43: A proposed architecture for cross-domain object detection. Blue elements represent standard Faster-RCNN components, purple elements - domain adaptation components and yellow elements are Faster-RCNN modified components for continual learning, which will be discussed later.

The model of Chen et al. [22] was one of the first models to use a two-stage alignment, which was discussed earlier in the Adversarial feature learning section. Xu et al. [92] expanded further on this method by experimenting with categorical regularization. Upon analyzing multiple papers, Xu et al. comments that aligning only image-level features would result in the alignment of the non-transferable background regions between the source and the target domains, thus leading to poor accuracy in the target domain [92]. Instance-level alignment would allow to reduce the distance between the local features, such as object appearance, size and viewpoint [22]. Naturally, it is not possible to omit image-level alignment as it is important to address adaptation of the global features of the image, such as image scales, illumination and style [22].

Therefore, this setup employs a mean teacher based Faster-RCNN [12] setup with two domain adaptation networks in the student model. Similarly to the preceding architectures [20, 73], data augmentation is used to assist in the training process, For these experiments, random cropping was excluded from tests as it was negatively affecting the pseudo-labeling. Instead, a few other transformations such as histogram equalizing and sharpness adjustments, were added to the strong augmentations list. The complete list of used augmentations is shown in Figure 44. It is believed that random rotations and affine transformations are not as important with the T-LESS dataset as the number of images taken in a rotational manner seems to be fairly sufficient.

Identically to the earlier experiments, the backbone was chosen to be ResNet [45] with 101 layers. Following Li et al. [20], one adaptation module is attached to the output of a CNN backbone. Another domain adaptation network is attached to the output layers of Faster-RCNN. While the backbone generates image-level features, output layers of Faster-RCNN produce the foreground region proposals. Both modules utilize a standard GRL block [18] followed by a domain classifier.

In this work, the image-level domain classifier is similar to the basic CNN architecture illustrated in Figure 4 and it was defined in accordance with the setup of Li et al. [20]. It has three convolutional layers, batch normalization and leaky-ReLUs followed by a convolutional classier layer. The instance-level classifier resembles the architecture in [92] and consists of two linear layers with leaky-ReLUs and dropout layers and finalized with a classifier and a sigmoid. Dropout layers were added due to the reasons specified in Section 2.3.2. The pseudo-code of the proposed architecture can be found in the Appendices.

Naturally, both an image and a region proposed by Faster-RCNN should represent the same domain. Similarly to Xu et al., to minimize the difference between predictions of the domain classifiers, they are regularized using $\mathcal{L}_{\text{consist}}$ consistency loss. In these experiments, consistency loss is based on simple MSE loss between instance- and image-level domain classifier loss terms $\mathcal{L}_{\text{ins}}$ and $\mathcal{L}_{\text{img}}$, as it was described in Equation 2.

The calculations of both $\mathcal{L}_{\text{ins}}$ and $\mathcal{L}_{\text{img}}$ follow the practices outlined in Equation 5. Additionally, the consistency loss $\mathcal{L}_{\text{consist}}$ follows the formula in Equation 6. However, instead of L2 norm, MSE is utilized.

To trade-off between different loss parameters, the weights of the domain classifi-

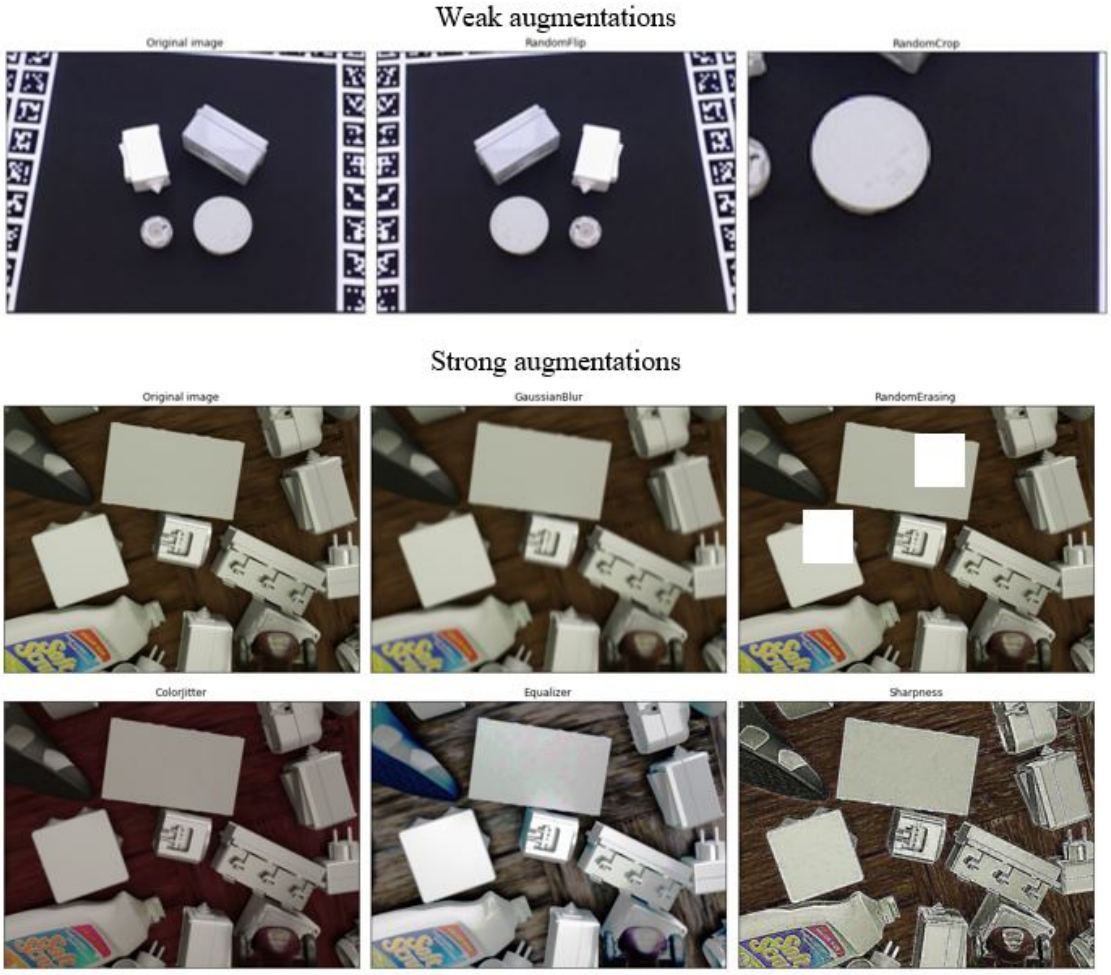


Figure 44: Augmentations used in the experiments.

cation and consistency losses have been added to the original setup of the Adaptive Teacher network. The complete formula for loss computation is displayed in Equation 11.

$$\mathcal{L} = \mathcal{L}_{\text{sup}} + \lambda_{\text{unsup}} \cdot \mathcal{L}_{\text{unsup}} + \lambda_{\text{img}} \cdot \mathcal{L}_{\text{img}} + \lambda_{\text{ins}} \cdot \mathcal{L}_{\text{ins}} + \lambda_{\text{consist}} \cdot \mathcal{L}_{\text{consist}} , \quad (11)$$

where $\mathcal{L}_{\text{sup}}$ is the standard Faster-RCNN detector loss in the student model; $\mathcal{L}_{\text{unsup}}$ is the unsupervised loss between pseudo-labels generated by the teacher model and supervised predictions and the λ-s are the corresponding trade-off parameters to shift the weights of the loss terms. The $\mathcal{L}_{\text{sup}}$ is calculated in a similar manner as it was introduced earlier in Equation 3.

Pseudo-labels are obtained identically as in the original Adaptive teacher imple-

mentation [20]. The noisy pseudo-labels are filtered using a confidence threshold and a class-wise NMS [73]. These pseudo-labels are used as ground-truth in the calculation of the unsupervised loss $\mathcal{L}_{\text{unsup}}$. The unsupervised loss is essentially based on a standard Faster-RCNN loss. However, it does not include the bounding box regression loss, as it not possible to filter out the bounding boxes using only confidence threshold, which was defined by Liu et al. [73]. To improve the quality of pseudo-labels, EMA is applied to gradually move weights from the student to the teacher model as discussed in the Mean teacher training section, and it is the only place where updating the teacher model takes place [20].

Multiple experiments have been conducted with the architecture defined in Figure 43. These experiments will be discussed in the Results section. In the following section, a proposed continuously learning domain-adaptive object detection setup is introduced.

3.5 Object detection with continual learning

Although the pre-trained model performs fairly well on the target dataset, it will not be able to predict unknown objects. To address the scalability of the model, which was defined in the Thesis objective, the proposed architecture should be able to keep learning when new data is added. Often such problems are addressed by a special case of transfer learning known as "fine-tuning". A typical example of fine-tuning is done by retraining the top level components of the network. This procedure would enable the network to apply previously learned knowledge on an unknown yet similar problem, while significantly reducing the training time. In regards to the objectives of this thesis, it would allow to utilize the knowledge of the previously learned equipment parts to promptly retrain it and apply it to new classes of objects. In the first set of experiments, a naive fine-tuning approach was attempted. The model was trained based on the architecture, which was demonstated in the Regularized Cross-Domain Adaptive Teacher. The regular training stage included both source and target datasets that contained classes 1 to 20, while the fine-tuning stage only included the data for classes 21-30 and the pre-trained weights file from the previous stage. After the training has been carried out for the first 20 classes, the FC layers of the trained model have been removed and replaced with new FC layers that can produce predictions for 30 classes. Both models were trained for the same number of iterations. The results can be found in Table 6 below.

Table 6: The naive fine-tuning results.

	Model trained on data from classes 1-20	Model trained on data from classes 21-30
Base learning rate	0.001	0.0005
Outputs	Predictions for 20 classes	Predictions for 30 classes
Max iterations	35000	35000
Training time	about 8 hours	about 8 hours
Data	source (labeled) and target (unlabeled) datasets limited to classes 1-20	Saved weights(.pth) from the original model(1-20) + datasets limited to classes 21-30
Results	**AP** 58.1596, **AP50** 75.27, **AP75** 66.89	**AP** 19.93, **AP50** 29.78, **AP75** 22.36

As can be easily noted, the results dropped significantly even though the initial

weights are supposed to give a decent advantage at the start. This setup would not only have low performance on the new data, but also unable to classify and localize original dataset. A few important modifications were considered in the next experiment:

1. In order to fine-tune the network successfully, a freezing process should be carried out on the deeper, more specifically, on the backbone layers of the network. Skipping this step is the main result of the performance drop in Table 6. By freezing deeper parts of the network, key trained features will be preserved, and the number of trainable parameters will be reduced, which will in turn speed up the training time.

2. Too many unfamiliar cases will lead to a higher performance drop. Adding fewer additional classes in each subsequent task will help to maintain the performance drop at the lower level. This has been proven by experiments of Peng et al. [93]. In other words, instead of retraining the original model with 10 more unknown classes, the model should be retrained with one or two at most.

3. Additionally, the weights file `.pth`, which was originally trained for the first N classes, also contained the states of the optimizer, i.e. of the training process. This included the iteration number and the learning rate at that iteration. As a result, the training process starts from a much lower base LR, thus decreasing the overall training speed. Therefore, before proceeding further, the optimizer states were removed from the dictionary of the weights file.

4. Finally, instead of replacing the original FC layers altogether, a new architecture is proposed, as illustrated in Figure 45 (b).

Following the studies presented in the Continual learning section, the detector head of the custom model is additionally modified to address a scalable training setup. The architecture presented in Figure 45 (b) replicates the method proposed by Rusu et al. [81], but applies it in a cross-domain mean teacher setup. An overview of this method was shown earlier in Figure 43, where both detector heads of the student and the teacher networks are modified. In this experiment, the original detector head is frozen, which includes both the classifier and the box regressor. Freezing essentially prevents the weights from being modified by disabling the learning rate updates. Consequently, new output layers are attached to the network. These layers are responsible for the unknown M number of classes and unlike the original detector, they are not frozen. The original detector head classifies an $N + 1$ number of objects, which also includes the background. Therefore, to preserve data integrity, the appended detector head ignores the background class and only predicts M new objects.

One potential disadvantage to such approach is that the original detector head is trained to recognize the background as well, which also included the shapes of the new M objects. Moreover, the weights for the original detector are frozen. Therefore, it is believed that the appended detector head will struggle predict the new classes if the original detector forces it to be predicted as "background".

To solve this problem, a new architecture has been introduced in Figure 45 (c). This model further combines the ideas proposed by Rusu et al. [81] and Donahue et al. [80], which were introduced in the Continual learning section. Instead of completely freezing the original detector weights, the learning rate is regularized by reducing the base learning rate for the previously trained classifier. The extended FC layers are trained with a standard LR. According to Parisi et al. [23], reduced learning rate will prevent the model from making significant changes in the original weights, while also learning new data. This might potentially enable the network to slowly be re-trained and learn the difference between the new classes and the background. Meanwhile, the dynamically expanding architecture of Rusu et al. [81] will focus on fighting the catastrophic forgetting problem. Ultimately, in the proposed method the model aims to learn new data using network expansion with regularization. The results of implementing such architecture are summarized in the Continual learning results section.

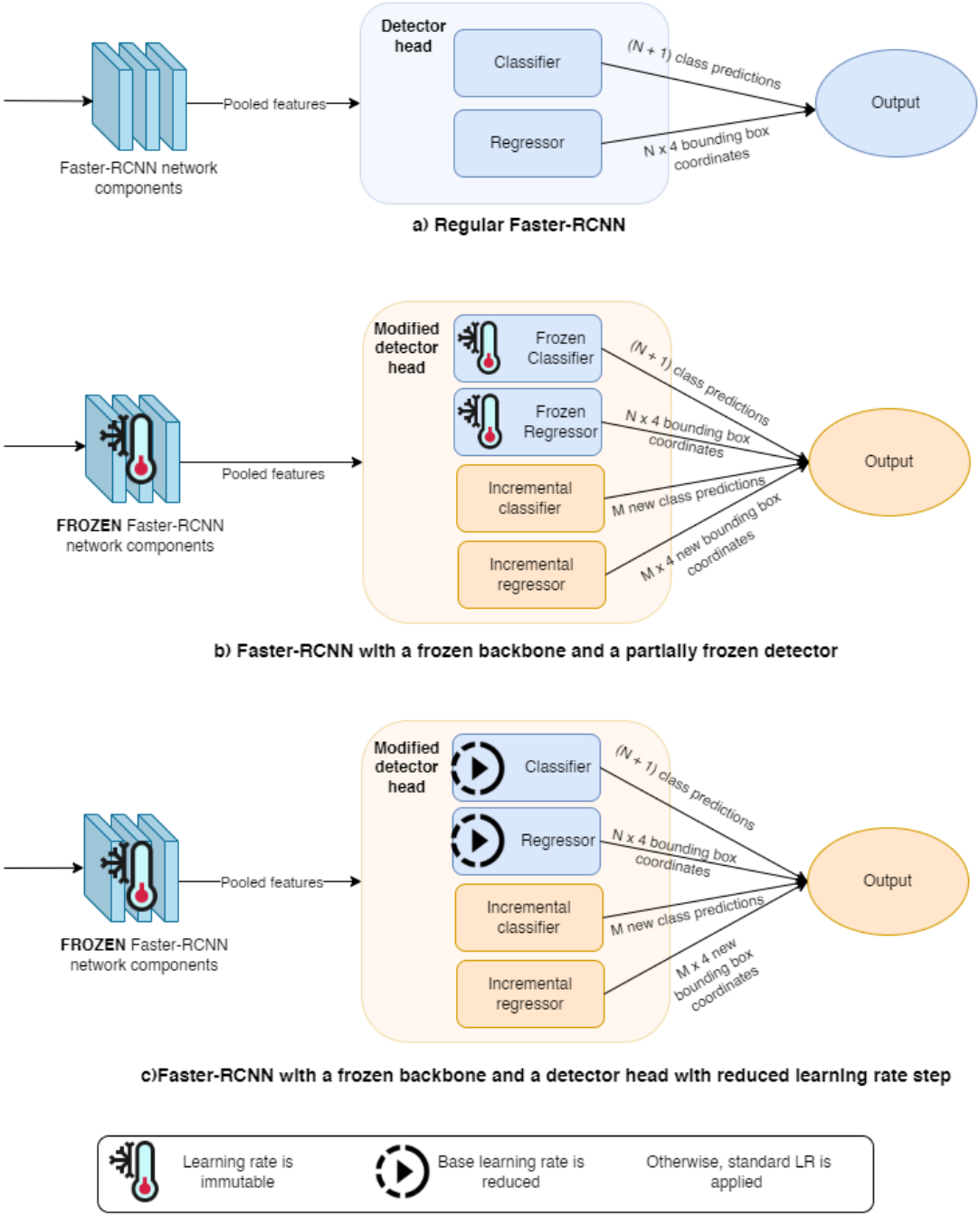


Figure 45: A proposed architecture for continual learning setup. Blue elements represent standard Faster-RCNN components, yellow elements are modified components for continual learning.

4 Results

This section summarizes the key findings of the experiments that were conducted based on the architectures presented in the sections 3.3, 3.4 and 3.5 of the thesis. Additionally, this section briefly introduces the simplistic UI and evaluates the proposed model on one equipment item from a real plant.

4.1 Cross-domain adaptation results

In order to achieve fair comparison, all experiments obey the following specifications:

1. The results presented in the subsequent sections will be evaluated predominantly on the T-LESS dataset as introduced in the Dataset section.
2. Thirty objects in the dataset were given arbitrary class names using the following notation: `Model 1..Model 30`
3. According to the setup defined in Figure 43, the teacher model accepts only the weakly-augmented images from the target domain. The primary objective of the model is to improve performance on the target images. Therefore, the primary objective for this ensembled model is to improve the `AP` on the teacher model. Thus, `AP50` metrics are recorded for the target images and are then used for comparison.
4. The base learning rate is defined as `BASE_LR` = 0.001.
5. The number of iterations is fixed at `MAX_ITER` = 50 000.
6. The evaluation happens every `EVAL_PERIOD` = 1000 iterations. Considering that there are 50 000 iterations total, this results in 50 evaluation points across the `AP50` plot.

4.1.1 Scheduler adjustment

First set of experiments was conducted in accordance with the Experiments with Adaptive Teacher section. For the baseline comparison, the Adaptive Teacher network was initially trained as it is. This implies that no external components were added and the original learning rate scheduler was used. The pattern of the original scheduler was presented earlier in Figure 42 (a). The results of such setup are presented in Figure 46. The first graph illustrates the evaluation `AP50` results for both the teacher and the student models. Although the training process seems idle for the first 20 000 iterations, the model eventually finds better gradients and stabilizes at max (`AP50`) = 71.40 % at 31 000 iterations. Even though it might seem that the loss value is decreasing by the end of the training, the `AP50` value also steadily drops. In order to address the noisy predictions shown earlier in Figure 41, the learning rate scheduler has been modified to implement a cosine annealing algorithm without restarts [91], as shown in Figure 42 (b). The original warm-up period of `WARMUP_ITERS` = 1000 is preserved.

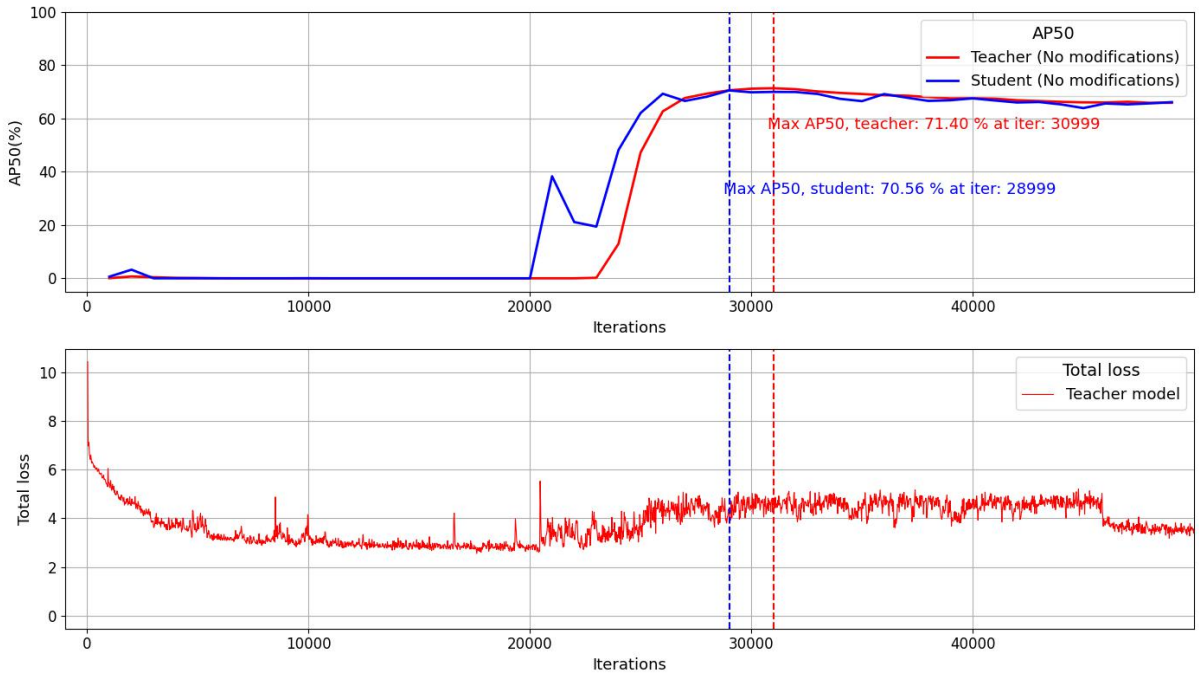


Figure 46: Results of the original Adaptive Teacher model evaluated on the custom T-LESS dataset without any modifications.

Theoretically, this setup would have allowed to find better gradients and reduce overshooting. The results achieved after modifying the scheduler are shown in Figure 47. Although the `AP50` value peaked much faster with the new scheduler, it was only able to reach max (`AP50`) = 62.50 %, which is almost 10 % lower than the original result.

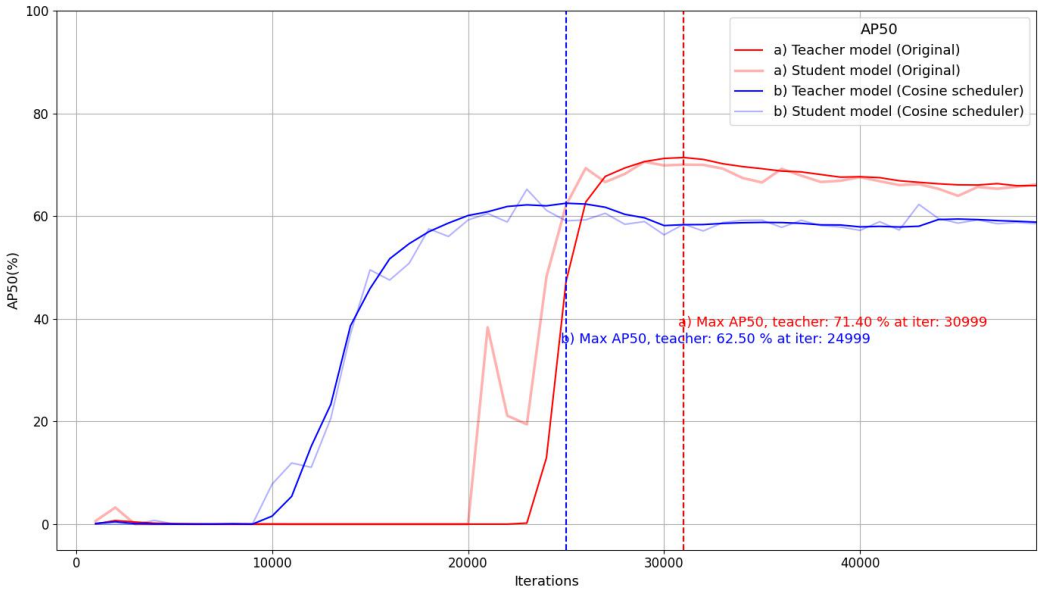


Figure 47: a) `AP50` results of the original model and b) `AP50` results of the model with a cosine scheduler.

4.1.2 Additional augmentations

Following the ideas proposed in Figure 44, an identical model was trained with two additional strong augmentations. This in theory allows to further diversify the dataset, which will make it less prone to over-fitting. Furthermore, in this and the subsequent experiments, the model leverages the early stopping algorithm with the parameter of `PATIENCE` = 10. The early stopping algorithm will prevent the model from unnecessary training in case if the `AP50` does not improve for more than `PATIENCE` × `EVAL_PERIOD` = 10000 iterations. The results of the evaluation process with the additional augmentations can be found in Figure 48.

By analyzing these results, it can be easily noticed that the model reaches the peak much faster and the maximum value (`AP50` = 70.53 %) is higher than the equivalent model without additional augmentations (`AP50` = 62.49 %). Additionally, this model achieves the results, which are competitive to the original Adaptive Teacher implementation (71.40 %).

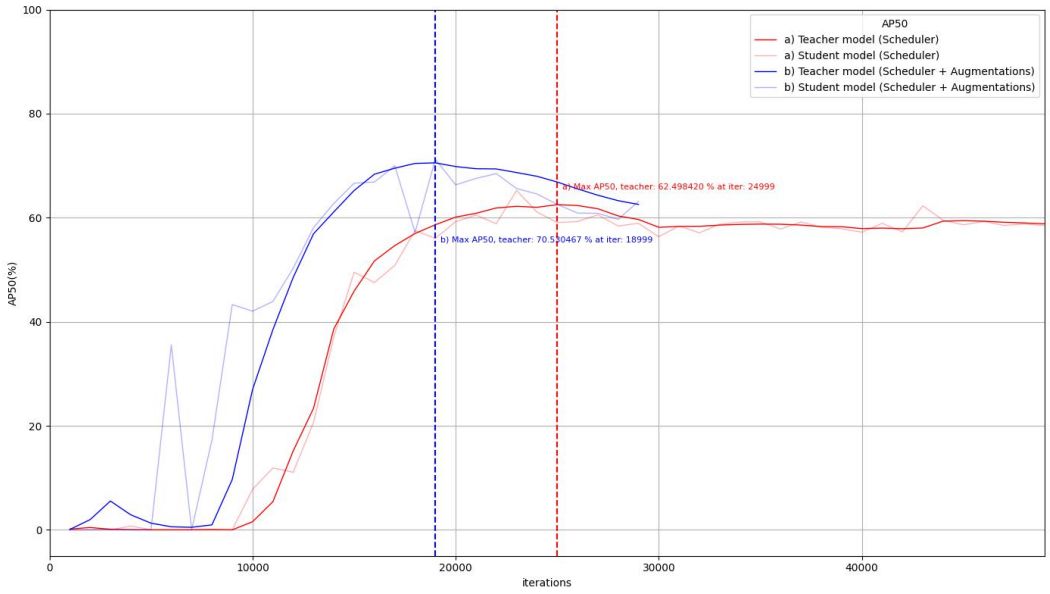


Figure 48: a) `AP50` of the model with a cosine scheduler only and b) `AP50` results of the model with a cosine scheduler, two additional strong augmentations and the early-stopping algorithm.

4.1.3 Instance-level DA and consistency regularization

The following set of experiments evaluates the custom model presented in Figure 43. The model utilizes the same principles as in the previous experiment, which includes the cosine scheduler and additional augmentations. On top of it, in this experiment, an instance-level domain adaptation is added along with the consistency regularization term. The regularization weights from Equation 11 are initialized as $\lambda_{\text{consist}} = \lambda_{\text{ins}} = 0.07$. Figure 49 presents the comparison between this model and the model presented in Section 4.1.2.

As it can be concluded, the plot with consistency regularization follows a similar pattern as the pattern in the original model with two custom augmentations. However, the results drop from max (`AP50`) = 70.53 % back to max (`AP50`) = 62.87 %. In order to verify the design of the custom components and that they work as intended, an additional experiment was conducted.

Figure 50 illustrates the plots of the instance-level loss and the consistency loss. As it was defined earlier in the Regularized Cross-Domain Adaptive Teacher section, the proposed model, similarly as any other adversarial DA model, aims to maximize the instance-level alignment loss in order to confuse the classifier and produce domain invariant features. On the other hand, the model also aims to minimize the consistency loss to force both instance- and image-level classifiers to generate identical outputs for the same image. Although both terms are working as

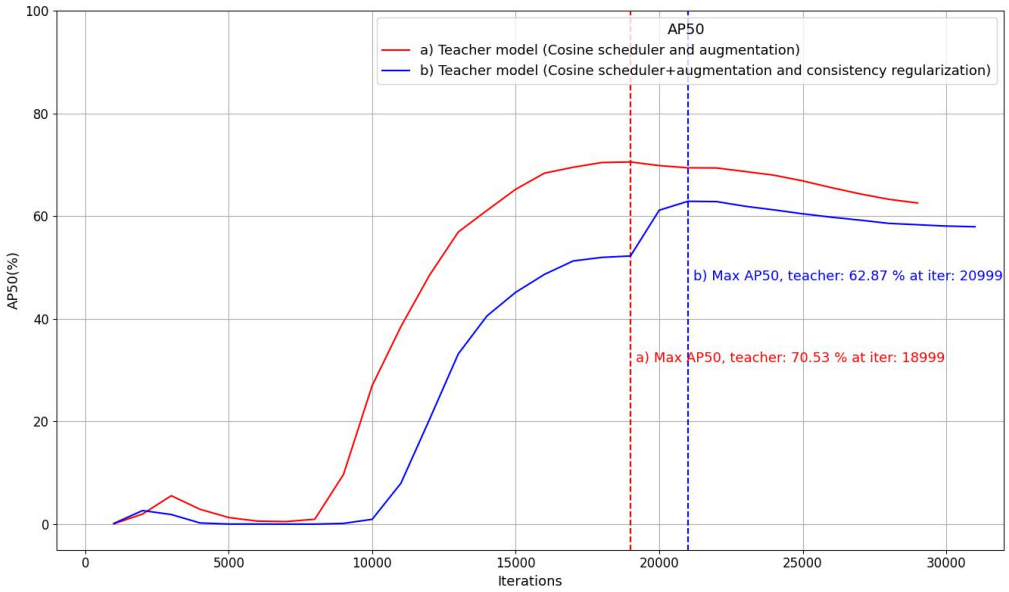


Figure 49: a) `AP50` of the model with a cosine scheduler and two extra augmentations against b) `AP50` of the model with a cosine scheduler, two additional augmentations and a consistency regularization.

designed, which can be concluded from Figure 50, the `AP50` value does not improve significantly compared to the original Adaptive Teacher model.

A small-scale set of experiments has been additionally carried out in order to identify the optimal weights λ_{consist} and λ_{ins}. In order to facilitate the training speed, while preserving fairness in the comparison process, the maximum number of iterations in all experiments was set as `MAX_ITER` = 30 000. Additionally, to decrease the training time, only the classes 1 to 20 were used for training. The results of six different experiments are shown in Figure 51.

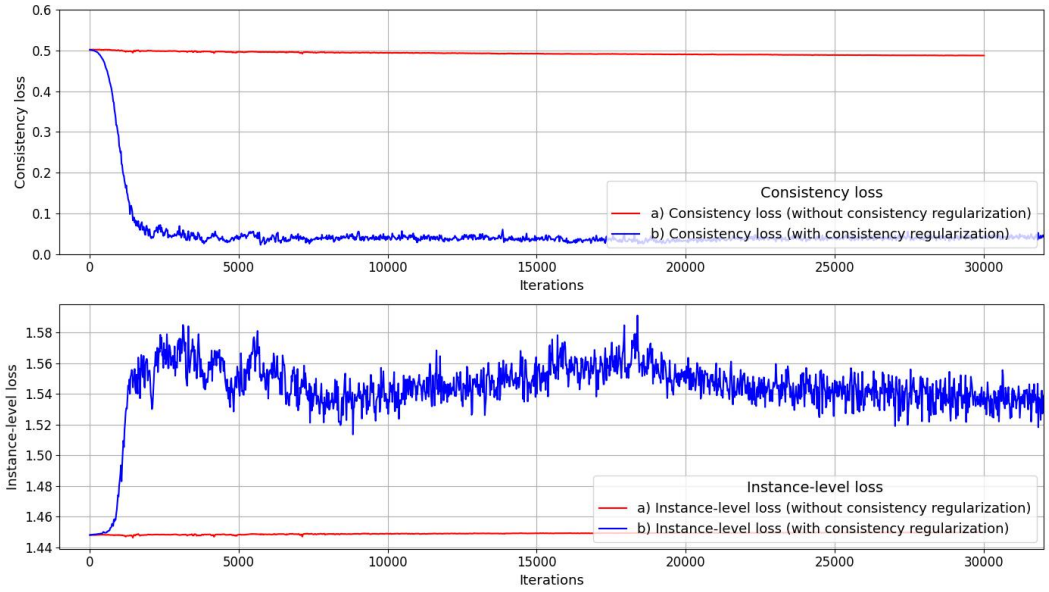


Figure 50: a) The total loss calculation is independent from consistency loss and the instance-level loss terms b) The total loss is proportional to the consistency loss and the instance-level loss terms.

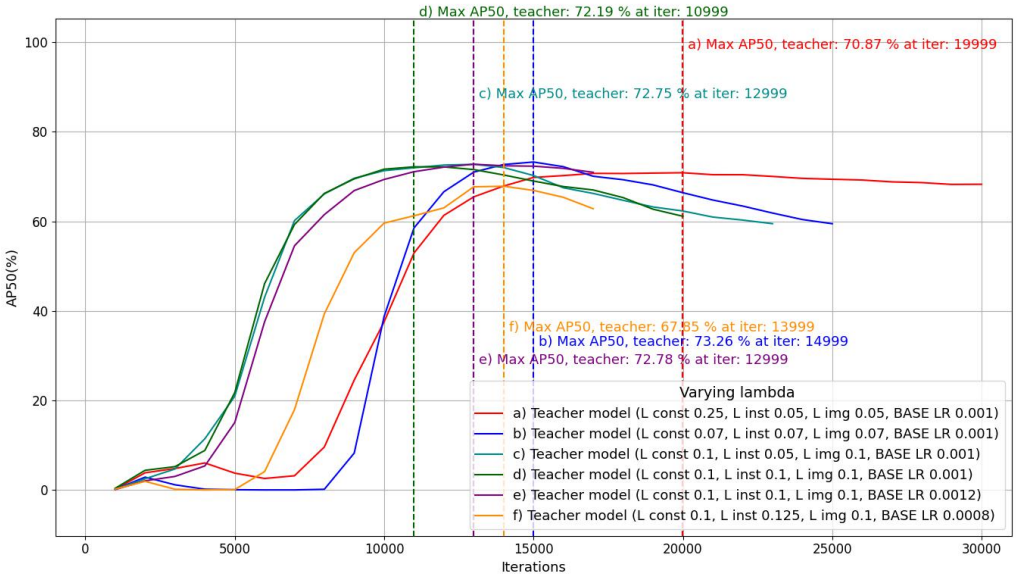


Figure 51: `AP50` values for varying weight parameters of λ_{consist} , λ_{ins} and `BASE_LR`.

From these results, the plot with the $\lambda_{\text{consist}} = 0.07$, $\lambda_{\text{ins}} = 0.07$ and `BASE_LR` = 0.001 were identified to be the best parameters with the resulted `AP50` = 73.26 %. However, in practice, more comprehensive experiments should be carried out to determine the best trade-off parameters and the learning rate.

To verify the performance of the components, one last experiment evaluates the network without the cosine scheduler, which seemed to negatively affect the performance of the subsequent experiments the most (see Figure 47). The results of the Regularized Cross-Domain Adaptive Teacher model with the original scheduler are presented in Figure 52.

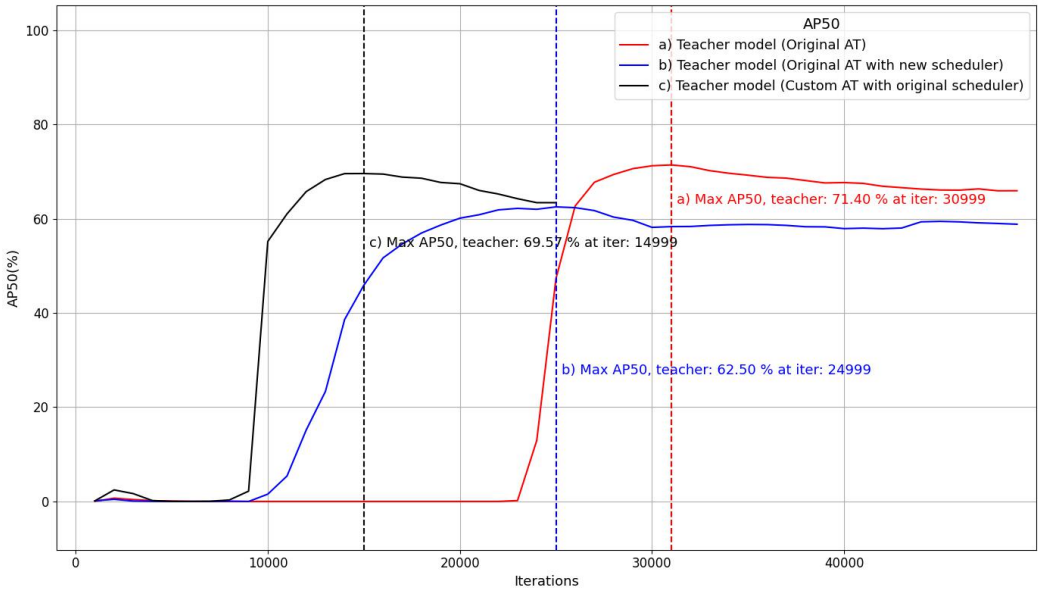


Figure 52: a) Original model without any modifications b) The custom model with the cosine scheduler c) The custom model with the original scheduler.

According to the results from Figure 52, the model with consistency regularization, custom augmentations and the default scheduler shows competitive results (`AP50` = 69.57 %) compared to the original Adaptive Teacher model (`AP50` = 70.56 %, while also peaking significantly earlier (>15 000 iterations faster).

An additional step has been carried out to verify whether a higher distribution of certain classes affects the model performance (see Figure 38). The results of the custom Adaptive teacher model with the original scheduler were filtered and averaged out for the first four classes `Model 1..Model 4`, which were then compared to the average of the remaining classes `Model 5..Model 30`. The results can be found in Figure 53.

The results of `AP50` = 53.65 % for the classes `Model 1..Model 4` seem to be significantly lower than `AP50` = 71.32 % for the remaining classes in the identical setup. This might indicate that higher object distribution in the dataset results in

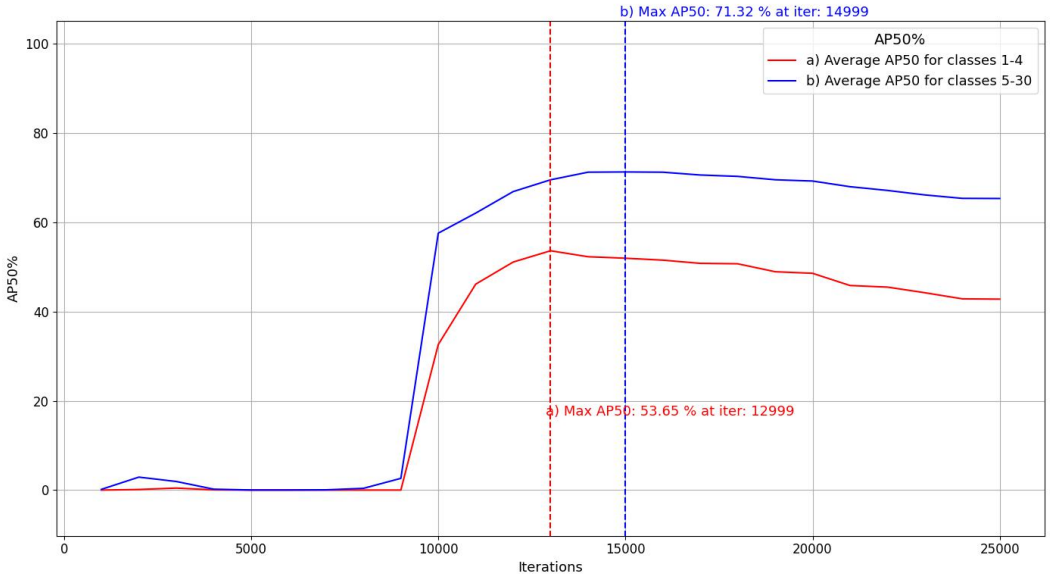


Figure 53: a) Average values of `AP50` for the classes 1 to 4 b) Average values of `AP50` for the classes 5 to 30 extracted from the same model results.

lower detection accuracy as the model struggles to generalize. However, due to a low number of samples, these results might also suggest that the first classes `Model 1..Model 4` are simply harder to detect. Therefore, more experiments are needed with higher variety in the dataset.

4.2 Continual learning results

As was introduced in Figure 37, the entire T-LESS dataset contains 30 different models with objects named arbitrary as `Model 1..Model 30`. To evaluate the methodology presented in the Object detection with continual learning section, the following procedure was applied:

1. In the initial experiment, the network was trained on the classes `Model 1..Model 30`.
2. The second network was only trained on the class `Model N`, where `N` is a number between 21 and 30.
3. The third network was trained on the classes `Model 1..Model 20`.
4. The third network was additionally re-trained to predict the class `Model N` in a continuous manner from a pre-trained model that was derived in Step 3.

All networks were based on the model with the cosine annealing scheduler, two custom augmentations and the consistency regularization term. This model was presented earlier in Figure 49.

Training on the single-class `Model 21` was selected for the first set of experiments. The results on the class `Model 21` were filtered out, extracted and, ultimately, the performance was compared between the three suggested methods of training: training from scratch, training individually on one class and training continuously. The combined results for the single class `Model 21` are presented in Figure 54.

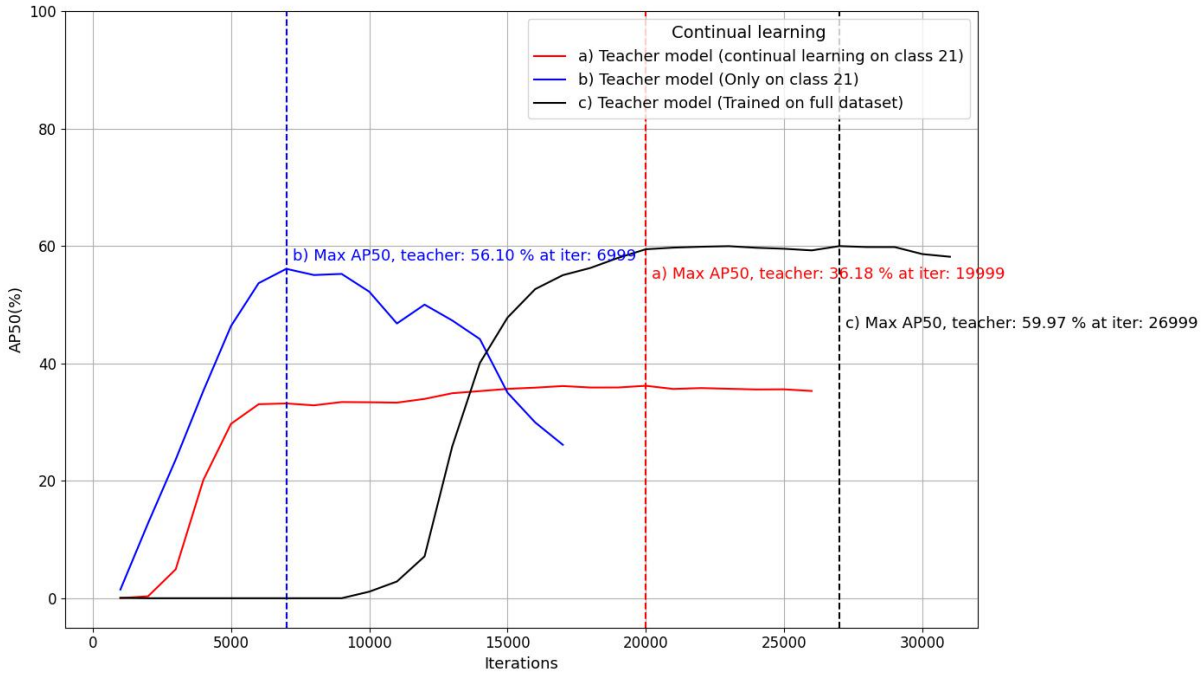


Figure 54: The `AP50` results for class `Model 21` evaluated in three different setups.

The model trained individually on the `Model 21` (Figure 54, a) rises just as rapidly as it declines. This is contrary to the model trained on the entire dataset (Figure 54, b), which takes longer to train but also grants better performance on the given class. Meanwhile, the model trained in a continual manner (Figure 54, c) learns the new class almost just as fast as the model trained purely on the `Model 21`. However, after a while it stagnates and barely reaches 36%.

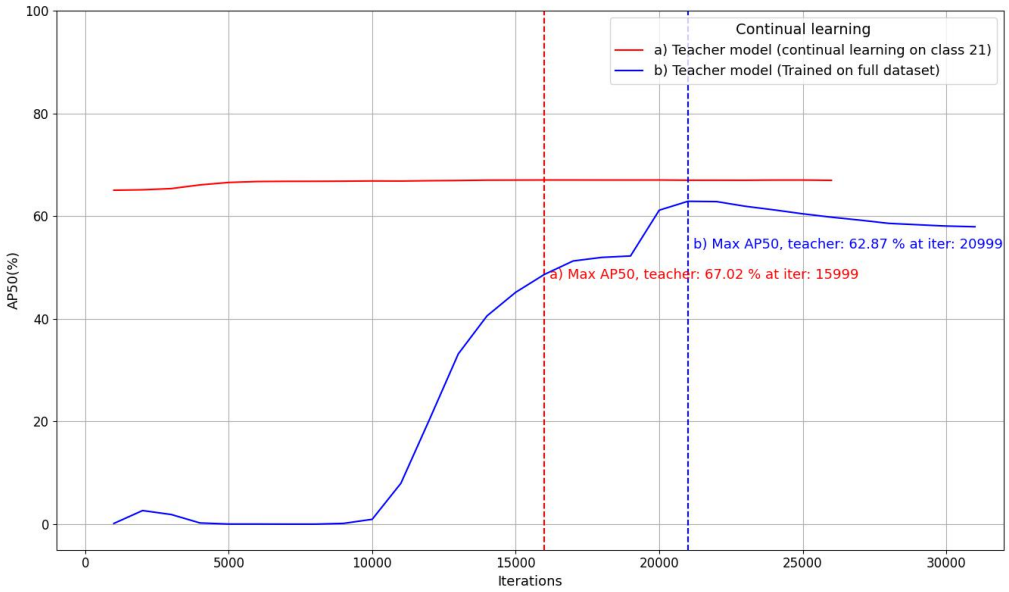


Figure 55: a) The total `AP50` value evaluated on the classes `Model 1..Model 21` using continual learning b) The total `AP50` results for the classes `Model 1..Model 30` on the model that was trained from scratch.

The average continual learning results of the total `AP50` for classes `Model 1..Model 21` (Step 4) were also compared to the average total `AP50` results for classes `Model 1..Model 21` when trained from scratch (Step 1). This comparison is presented in Figure 55. According to this plot, the overall `AP50` trend seems to be increasing, which means that the model is slowly learning new data without losing the previously obtained knowledge, thus solving the catastrophic forgetting problem.

The plots follow similar patterns when repeating the training steps for other classes of objects. The results can be found in Figure 56.

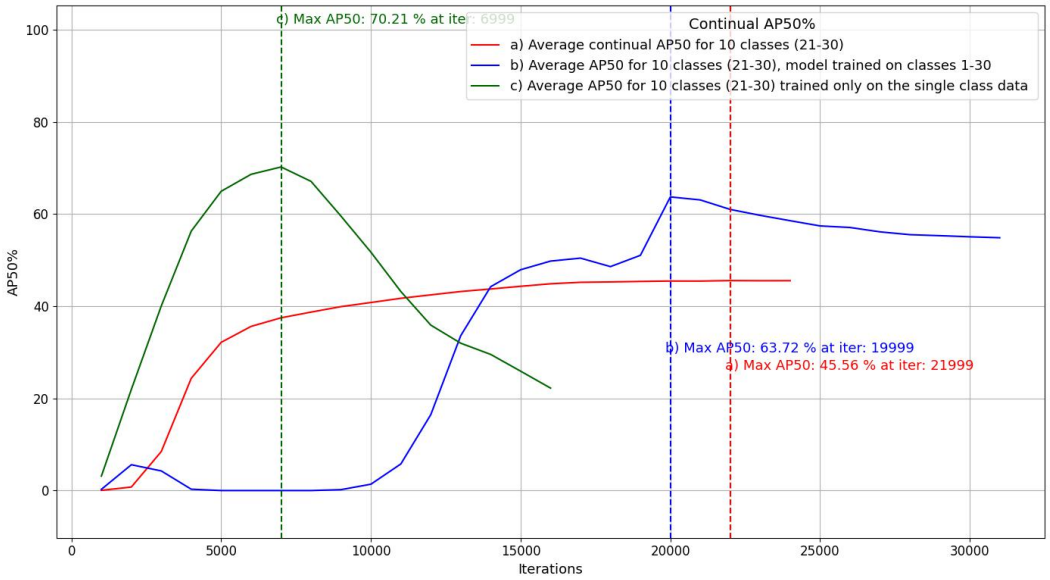


Figure 56: a) The average `AP50` value for continual learning on classes `Model 21..Model 30` given the model trained on classes `Model 1..Model 20` b) The average `AP50` value for classes `Model 21..Model 30` extracted from the original model trained from scratch on classes `Model 1..Model 30` c) The `AP50` value for each of the classes `Model 21..Model 30` when trained individually. The values are then averaged out for all 10 classes.

Here, Step 4 was repeated for the classes `Model 21..Model 30` based on the model derived from Step 2. Consequently, the average of the mean results for each of the classes were combined to form the plot in Figure 56 (a). For comparison, the training results for the classes `Model 21..Model 30` were extracted from the results of the training on the entire dataset (Step 1). Such plot is illustrated in Figure 56 (b). Finally, Figure 56 (c) illustrates the `AP50` value for the average of each of the classes `Model 21..Model 30` when trained individually (Step 2).

As it can be concluded, training the model on each of the classes individually yields the best results `AP50` = 70.21 %. Additionally, it can be noted that the model achieves a higher `AP50` when trained from the scratch (`AP50` = 63.672%), compared to the model trained continuously, which saturates at the much lower value of `AP50` = 43.56%. However, it takes considerably less time to train the model continuously, as this lifelong learning model reaches 90 % of its maximum `AP50` value already in about 7000 iterations.

To conclude, the presented continual learning approach has proven to be able to learn new classes. Despite being less efficient that training the model from scratch, it adequately solves the catastrophic forgetting problem, which was discussed earlier in Section 2.8.

4.3 Deployment results

In order to showcase the performance of the proposed model, a simple web app was developed. The app was hosted on a local server and it leverages Flask API to connect the detector app to the user interface. The app utilizes the model, which was presented in Figure 52 due to its relatively good performance and the fastest training speed among the competitors. The prototype of the UI is presented in Figure 57.

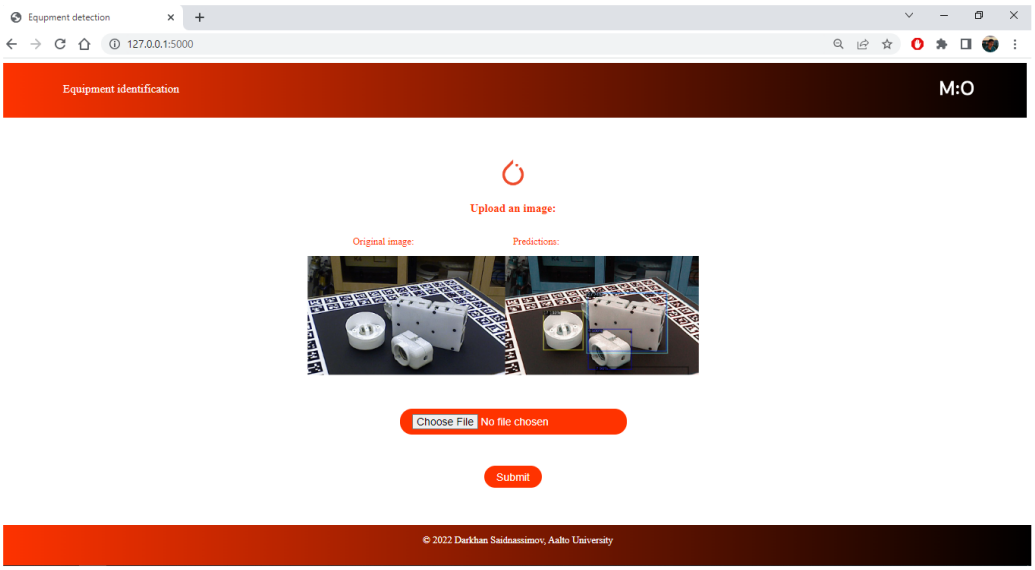


Figure 57: A screenshot of the simple web app. The image on the left-hand side represents an uploaded target image with the objects to predict, and the image on the right-hand side returns the localized objects.

The app sends `POST` requests to upload an image to localhost via Flask API. The uploaded image is forwarded as an input to the model and returned to the user interface along with the predictions (if any) after running the inference on the CPU node. Loading an image to the model takes about 0.0012 s, while running the inference on the image takes about 0.0466 s. However, the uploading time is essentially the bottleneck of the detection process as it would vary depending on the internet connection speed and whether the server is located on the localhost or not.

The complete app is built using Python 3.9, while the model additionally utilizes packages and frameworks such as Pytorch 1.10.1, CUDA 11.3 and Detectron2 v0.6. The model was trained on Nvidia A100 units. The hardware and the computing resources were provided by CSC - Finnish IT Center for Science.

4.4 Evaluation on the real equipment

In the final set of experiments, the proposed model was briefly evaluated on the real equipment. As discussed in the Dataset section, the rendered images of the equipment can be obtained from the 3D CAD models. Although the images can be rendered using a script that would automatically rotate around the object in 3D-plane and save the images, for the purpose of this experiment, the rotation around the object was performed manually and the video recording of the operation was saved. Then, the recording was split into frames and 1 000 images of an industrial `HM-75S` pump were collected. These images were then labeled using LabelImg [**?**] annotation tool as illustrated in Figure 58. This subset of images will act as a source domain and is used only for training.

On the other hand, a handful of 28 real `HM-75S` pump images were collected from the plant. An example image is shown in Figure 59. These real images will act as a target domain and the main objective of this experiment is to identify and localize the real images of the pump correctly.

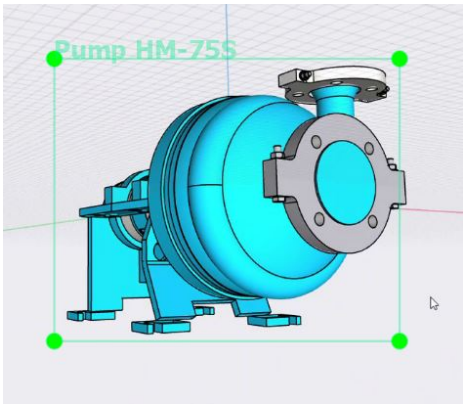


Figure 58: Labeled image of the rendered `HM-75S` pump

Figure 59: Unlabeled image of the real `HM-75S` pump

In theory, these images do not require labeling. However, as it was defined in the Dataset section, the target dataset is split into two subsets for training and for testing. This in practice means that in order to evaluate the model and to compare the results, the testing subset must be labeled. For this experiment, the splitting ratio was selected to be 70 % to 30 % for the training and testing subsets, instead of the originally proposed ratio of 85 % to 15 %. The number of the testing images was essentially increased to account for the target dataset of a fairly limited size. This resulted in 19 training and 9 testing images of the real pump.

The entire training dataset (1000 rendered and 19 real images) was then forwarded to the Adaptive Teacher model with consistency regularization and the original scheduler (see Figure 52). The training conditions were identical as in the previous experiments and the obtained results are shown in Figure 60.

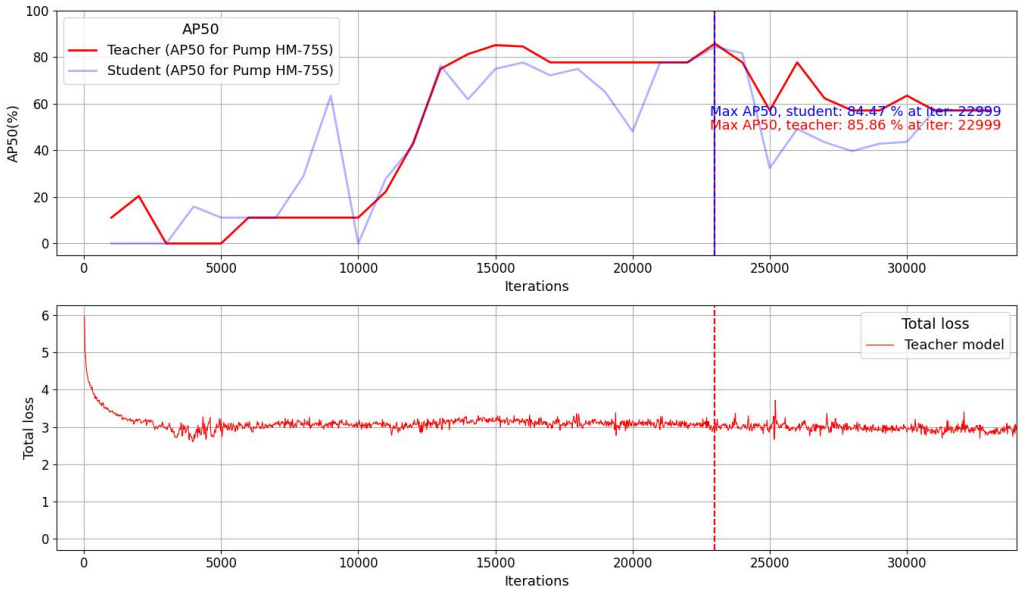


Figure 60: The performance of the proposed model on the custom dataset with one industrial object.

The results suggest that the model performance starts improving at 10 000 iterations, similarly to the earlier experiments (see Figure 52). The best `AP50` result of 85.86 % is achieved at 22 999 iterations. Finally, the model was deployed to the simplistic web app and the results of the prediction are visualized in Figure 61.

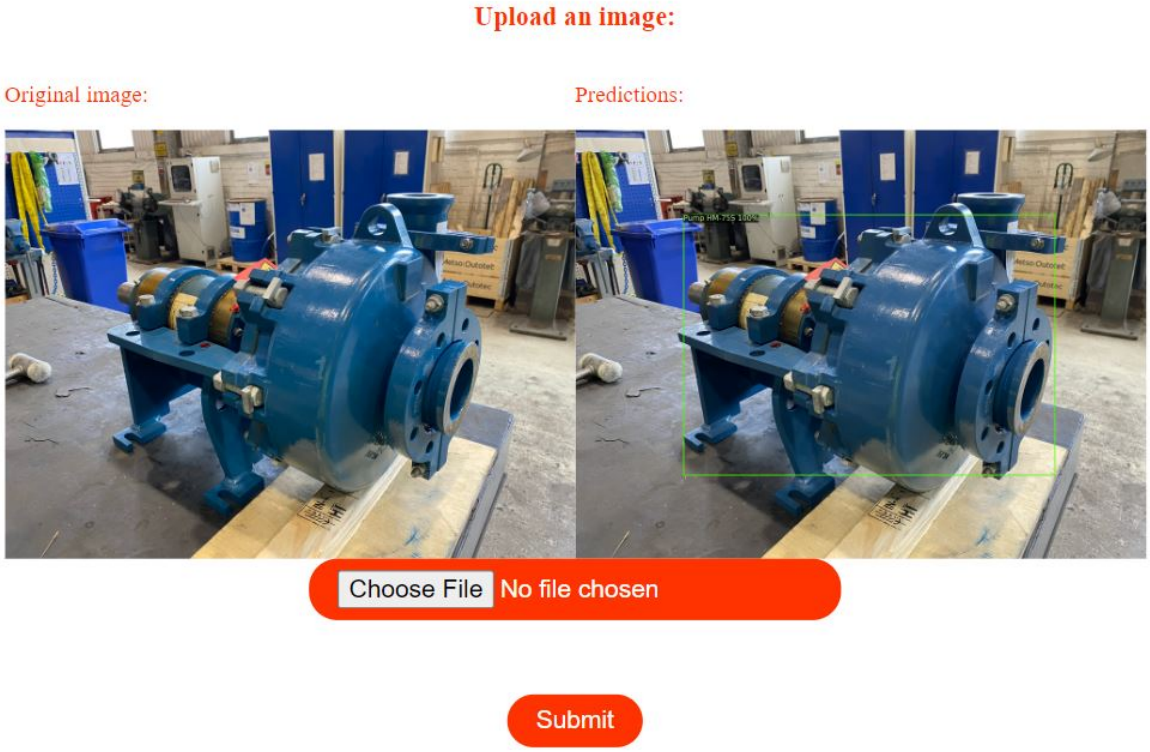


Figure 61: The demonstration of the model performance. The model was trained on the source and target images of the `HM-75S` pump.

Although the model seems to show a relatively good performance on the target dataset and correctly identifies the pump in the given images, this experiment was carried out to showcase the applicability of the method on the real-life industrial datasets, rather than to extensively evaluate the performance. In reality, a dataset of 28 real images is rather insufficient and a larger target dataset must be collected in order to produce more practical results.

5 Conclusion

This section summarizes the results of the thesis. First, the results of the proposed cross-domain object detection approach are presented. Next, the results of the introduced continual learning approach are outlined. Finally, this section discusses the directions for future work. The codebase for this project will shortly be published to https://github.com/darkhan-s/master-thesis-equipment-detection. However, the access will be restricted and will be available to certain parties upon request.

5.1 Summary of the DA results

This thesis evaluates the performance of multiple object detection approaches on a custom T-LESS dataset, which includes images of thirty industry-relevant equipment parts in two different domains. The source domain consists of the rendered labeled images from a simulated environment, while the target domain contains a smaller amount of real unlabeled images. The performance results after running the tests with the standard Faster-RCNN object detector suggested that the distribution gap between the domains exists, which leads to a significant performance drop.

In order to address the problem of domain shift, several state-of-the-art domain adaptation methods were suggested. Upon analyzing these methods, this thesis proposed a novel cross-domain object detection architecture. The architecture is primarily based on the Adaptive Teacher implementation, which aims to align two domains by means of mean teacher training, adversarial feature learning and pseudo-labeling. In this thesis, multiple versions of the Adaptive Teacher model have been evaluated. First, the original LR scheduler has been replaced with the cosine annealing LR scheduler without restarts. Consequently, additional augmentation techniques have been evaluated. Finally, inspired by the existing two-level adversarial feature learning approaches, the Adaptive Teacher model has been modified to include two additional components - instance-level domain alignment and consistency loss. Instance-level alignment was used to produce domain invariant features in the proposed regions of the Faster-RCNN network, which is the base detector of the Adaptive Teacher model. On the other hand, the consistency loss term was used to minimize the error between the outputs of the image- and instance-level alignment losses. The discussed modifications have been extensively evaluated on the custom T-LESS dataset using `AP50`, which is based on the traditional PASCAL VOC 2012 metrics. The comparison of different evaluated methods are presented in Table 7.

The results suggest that the original Adaptive Teacher model performs the best with max (`AP50`) = 71.40 %, followed by the original model with the cosine scheduler and two additional augmentations(max (`AP50`) = 70.53 %). On the other hand, the proposed mean teacher architecture with instance-level alignment and consistency regularization achieves competitive results (max (`AP50`) = 69.57 %) almost twice as fast compared to the original model. The optimal values of the trade-off λ parameters has been identified as $\lambda_{\text{consist}} = \lambda_{\text{ins}} = \lambda_{\text{img}} = 0.07$. Nevertheless, additional experiments with the hyperparameters are needed to achieve better results.

Table 7: The summary of the DA methods evaluated in this thesis.

Method	**Best result, iteration**	**Best AP50, %**	**Notes**
Faster-RCNN [12] *	20 000	13.63 %	The model was trained on images from the source domain and evaluated on the images from the target domain
Cycle-GAN [70] *	N/A	N/A	No metrics were recorded due to poor performance
D-Adapt [87] *	16 000	64.32 %	
Adaptive Teacher (AT) [20] without modifications	30 999	71.40 %	
AT with cosine scheduler	24 999	62.50 %	Cosine scheduler only
AT with extra augmentations	18.999	70.53 %	Cosine scheduler + Augmentations
AT with instance-level adaptation and consistency regularization (= **Custom AT**)	20 999	62.87 %	Cosine scheduler + Augmentations + consistency regularization with $\lambda = 0.7$
Custom AT with original scheduler	14 999	69.57 %	Original scheduler + Augmentations + consistency regularization with $\lambda = 0.7$
Oracle (Faster-RCNN) [12] *	20 000	98.19 %	The model was trained on target images and evaluated on the target domain

Note: The methods were evaluated on the target dataset and the best `AP50` scores are compared. For the methods marked with *, only the latest score is recorded, which is achieved at the maximum number of iterations.

5.2 Summary of the continual learning results

In order to address the scalability problem and to facilitate learning new data continuously, this thesis has additionally proposed an approach that addresses lifelong learning. The detector head of the custom Adaptive Teacher model has been modified to solve the catastrophic forgetting problem, which happens when previous knowledge gets replaced by new data. The model attempts to learn new data in a lifelong manner using the continuous network expansion, where a new classifier and a regressor are attached to the original detector head every time a new class is added. Additionally, to preserve learned knowledge of the older classes, the learning rate of the original detector head modules is regularized to avoid the catastrophic forgetting phenomenon, while enabling the model to learn new patterns of data. Table 8 presents the summarized results of the described continual learning approach.

These results align with the results provided by Parisi et al. [23], where the

Table 8: The summary of the continual learning approach.

Method	**Best result, iteration**	**AP50**	**Notes**
Results for continual learning on one class			
Single class training	**6999**	56.10 %	Given the images for the class `Model 21` → predict `Model 21`.
Training from scratch	26999	**59.97** %	Given the images for the classes `Model 1..Model 30` → predict `Model 21`.
Continual learning	19999 (6999*)	36.18 %	Given the weights for the classes `Model 1..Model 20` → predict `Model 21` (continuously).
Average results for continual learning on 10 classes 21-30.			
Single class training	**6999**	70.21 %	Given the images for the classes `Model 21..Model 30` → predict `Model 21..Model 30`. The average results for the separate classes are combined..
Training from scratch	19999	**63.72 %**	Given the images for the class `Model 1..Model 30` → predict `Model 21..Model 30`. Results filtered and the average value among the classes `Model 21..Model 30` is collected.
Continual learning	21999 (6999*)	45.56 %	Given the images and weights for the classes `Model 1..Model 20` and images for the classes `Model 21..Model 30` → predict `Model 21..Model 30` (continuously).

Note: 90 % of the `AP50` value marked by * is achieved within about 7000 iterations.

authors claim that the efficiency of continual learning drops as the complexity of data increases. Although many of the existing methods provide the means to solve the catastrophic forgetting problem in image classification, less research has addressed more sophisticated cases such as object detection. Therefore, this thesis has attempted to utilize continuous learning in a domain-adaptive object detection setup. However, more research is needed in order to solve a lifelong learning problem in an industrial environment.

5.3 Directions for future work

Inspired by Oza et al. [64] and by the Thesis objective, this thesis studied several research directions in domain adaptation. Oza et al. summarized a few scenarios, where the cross-domain object detection was not researched as extensively. A few of these scenarios overlap with the thesis goals and these will be discussed in the following section.

5.3.1 Other applications and real world datasets

As it was discussed in Regularized Cross-Domain Adaptive Teacher section, common DA approaches in object detection address either autonomous navigation, or common day-to-day datasets and less work has been conducted on industrial datasets. Additionally, many of these datasets are barely suitable to reflect the domain gap between the real world and the simulations. This thesis attempts to reduce the knowledge gap by utilizing an industry-relevant dataset. However, the target dataset proposed in the Dataset section was collected in a typical household setup, where the environment has a stable lighting and background. Although a small-scale dataset of one equipment item was prepared to prove the applicability of the model in real-life scenarios, for future work it is advised to experiment with larger datasets of images from a real industrial plant.

5.3.2 Other detectors

As could be noted from the Domain-adaptive object detection section, Faster-RCNN is a de-facto object detector framework that is used in such problems. Oza et al. [64] suggested to experiment with single-stage object detectors such as YOLO [15], SSD [13], FCOS [17] and DETR [53]. In this thesis, two-stage Faster-RCNN was selected due to its simple plug-and-play compatibility with Detectron2 [21] and with the existing DA methods. Additionally, the robustness that single-stage detectors offer is not as critical in this thesis. However, there is a potential of extending the application to video-stream and identify the objects in real-time. As it was discussed in the Object detection section, single-stage detectors offer a significant speed improvement over Faster-RCNN. Upon analyzing the feasibility of implementing other detectors in Detectron2 [21] framework, it was identified that FCOS network has already been adapted for use in Detectron2. Hence, it is believed that among the detectors listed, the FCOS detector will be the most compatible with the model proposed in Figure 43.

5.3.3 Improved scheduler

This thesis has only evaluated the cosine annealing scheduler without restarts. One possible way to improve the results presented in the Scheduler adjustment could be achieved by implementing the restart cycles introduced in the original paper by Loschilov et al. [91].

5.3.4 Imbalanced classes

Although the classes were distributed fairly well in the source dataset, huge class imbalance was observed in the target T-LESS dataset, as it can be noted from Figure 38. The models 1 to 4 are over-represented and are the largest in the target dataset by a tremendous margin. This could be a potential reason for the performance drop as the disproportionately highly represented classes shifted the alignment of the domains in their favor. Even though the class-imbalance problem has been briefly analyzed in Figure 53, more experiments with higher class imbalance are needed.

Oza et al. proposed re-weighting the classes based on their frequency. However, as the target domain does not contain any labels, it becomes a challenging task to track the frequency [64]. Luckily, the proposed in the Regularized Cross-Domain Adaptive Teacher section cross-domain adaptation method leverages pseudo-labels. Therefore, re-weighting the classes can be carried out in stages as the pseudo-labeling accuracy improves.

On the other hand, it is possible to benefit from the class imbalance. Section 4.2 discusses objects that are harder to detect. Instead of trying to re-weight imbalanced classes, one could provide a higher number of image samples for the objects that can be classified as "difficult", while providing a regular number of samples for the objects that are easier to detect.

5.3.5 Multi-source adaptation

Another practical problem formulated by Oza et al. [64] is that often the images can be collected from multiple domains. In the scope of the thesis, this could imply collecting datasets from multiple plants with varying environmental conditions, such as rainy weather, snow, daylight and nighttime. These conditions will dramatically impact the detection results. Therefore, more experiments should be carried out before releasing such an application into use. One suggested approach to this problem could be carried out by using even more sophisticated augmentation algorithms such as the ones listed in [94] to simulate such environments.

5.3.6 Group-level alignment

As was discussed in the Adversarial feature learning section, using instance-level DA as it is might cause alignment of the remaining background features from the rendered source domain. Therefore, one possible direction for future work could include replacing instance-level alignment with group-level alignment using clustering by visual similarity, as proposed by Rezaeianaran et al. [67].

5.3.7 Continual learning

In Figure 56, the performance of the proposed continual learning approach has been analyzed for ten different classes and their average has been calculated. However, to obtain a better statistical average, more experiments have to be conducted on a dataset with a larger pool of objects.

Although the Object detection with continual learning section addressed the catastrophic forgetting problem in a scalable setup, the experiments were only conducted with well-established and possibly outdated methods. Moreover, according to Parisi et al. [23], current state-of-the-art continual learning models are still far from being able to learn new tasks as efficiently as biological systems and to this day it remains a challenging problem.

In the survey on transfer learning, Zhuang et al. [59] mentioned that TL is not always beneficial for new tasks in two given scenarios - when the difference between the domains is too large and when the similarities may be too misleading and instead cause a negative transfer. Analogously, continuous learning might experience similar problems when attempting to learn new data, which therefore requires further research.

Even though the model is able to adapt to the dynamically expanding dataset, the model will scale linearly as new classes are added. Within this thesis, the model was tested on up to 30 classes and the resulted model weight files consumed nearly 2GB of storage and a few linear layers appended will not significantly affect the already spacious file size. Nevertheless, for the sake of optimization and as new continual learning solutions emerge, it is important to evaluate them in the given setup and explore new possible solutions.

References

[1] N. M. Awalgaonkar, H. Zheng, and C. S. Gurciullo, "Deeva: A deep learning and iot based computer vision system to address safety and security of production sites in energy industry," Mar. 2020.

[2] N. Saidnassim, B. Abdikenov, R. Kelesbekov, M. T. Akhtar, and P. Jamwal, "Self-supervised visual transformers for breast cancer diagnosis," in *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 423–427, 2021.

[3] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," Apr. 2017.

[4] Y. Shan, W. F. Lu, and C. M. Chew, "Pixel and feature level based domain adaption for object detection in autonomous driving," Sept. 2018.

[5] M. Banf and G. Steinhagen, "Who supervises the supervisor? model monitoring in production using deep feature embeddings with applications to workpiece inspection," Jan. 2022.

[6] J. Wu, S. Tang, X. Li, and Y. Zhou, "Industrial equipment detection algorithm under complex working conditions based on roms r-cnn," *PLOS ONE*, vol. 17, p. e0266444, 04 2022.

[7] L. Malburg, M.-P. Rieder, R. Seiger, P. Klein, and R. Bergmann, "Object detection for smart factory processes by machine learning," *Procedia Computer Science*, vol. 184, pp. 581–588, 2021. The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[8] J. Kim, J. Hwang, S. Chi, and J. Seo, "Towards database-free vision-based monitoring on construction sites: A deep active learning approach," *Automation in Construction*, vol. 120, p. 103376, 12 2020.

[9] "Metso Outotec: Rocksense," 2022. https://www.mogroup.com/portfolio/rocksense/, last accessed on 2022-06-20.

[10] "Metso Outotec: MouldSense," 2022. https://www.mogroup.com/portfolio/mouldsense/, last accessed on 2022-06-20.

[11] "Metso Outotec: FrothSense," 2022. https://www.mogroup.com/portfolio/frothsense/, last accessed on 2022-06-20.

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," June 2015.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," Dec. 2015.

[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," Mar. 2017.

[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," June 2015.

[16] J. Zhang, J. Huang, Z. Luo, G. Zhang, and S. Lu, "Da-detr: Domain adaptive detection transformer by hybrid attention," Mar. 2021.

[17] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," Apr. 2019.

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research 2016, vol. 17, p. 1-35*, May 2015.

[19] T. Hodaň, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[20] Y.-J. Li, X. Dai, C.-Y. Ma, Y.-C. Liu, K. Chen, B. Wu, Z. He, K. Kitani, and P. Vajda, "Cross-domain adaptive teacher for object detection," Nov. 2021.

[21] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." https://github.com/facebookresearch/detectron2, 2019.

[22] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster r-cnn for object detection in the wild," Mar. 2018.

[23] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," Feb. 2018.

[24] D. Etiemble, "Technologies and computing paradigms: Beyond moore's law?," June 2022.

[25] T. Hwang, "Computational power and the social impact of artificial intelligence," Mar. 2018.

[26] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," Apr. 2021.

[27] B. Mehlig, *Machine Learning with Neural Networks*. Cambridge University Press, oct 2021.

[28] Z. Meng, Y. Hu, and C. Ancey, "Using a data driven approach to predict waves generated by gravity driven mass flows," *Water*, vol. 12, 02 2020.

[29] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "A comprehensive survey and performance analysis of activation functions in deep learning," Sept. 2021.

[30] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," Nov. 2015.

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-propagating Errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[32] A. Albarghouthi, "Introduction to neural network verification," Sept. 2021.

[33] M. Alber, I. Bello, B. Zoph, P.-J. Kindermans, P. Ramachandran, and Q. Le, "Backprop evolution," Aug. 2018.

[34] N. O. Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," *in Advances in Computer Vision Proceedings of the 2019 Computer Vision Conference (CVC). Springer Nature Switzerland AG, pp. 128-144*, Oct. 2019.

[35] "Computer Vision," June 2022. https://paperswithcode.com/area/computer-vision, last accessed on 2022-06-20.

[36] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[37] M. Z. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, M. Hasan, B. Essen, A. Awwal, and V. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, p. 292, 03 2019.

[38] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," Apr. 2016.

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," Sept. 2014.

[40] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[41] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," May 2014. https://cocodataset.org/, last accessed on 2022-06-30.

[42] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sept. 2014.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Dec. 2015.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[47] A. Rastogi, "Medium: Resnet50," 2022. https://blog.devgenius.io/resnet50-6b42934db431/, last accessed on 2022-06-20.

[48] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," Apr. 2021.

[49] A. Pacha, J. Hajič, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, 2018.

[50] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," Aug. 2017.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," Nov. 2013.

[52] R. Girshick, "Fast r-cnn," Apr. 2015.

[53] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," May 2020.

[54] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, 2013.

[55] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," May 2017.

[56] "Faster-RCNN," June 2022. https://paperswithcode.com/method/faster-r-cnn, last accessed on 2022-06-22.

[57] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," Dec. 2016.

[58] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," Apr. 2018.

[59] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," Nov. 2019.

[60] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[61] S. Sun, H. Shi, and Y. Wu, "A survey of multi-source domain adaptation," *Information Fusion*, vol. 24, pp. 84–92, 2015.

[62] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing, 2018, 312: 135-153*, Feb. 2018.

[63] Y. Zhang, "A survey of unsupervised domain adaptation for visual recognition," Dec. 2021.

[64] P. Oza, V. A. Sindagi, V. VS, and V. M. Patel, "Unsupervised domain adaptation of object detectors: A survey," May 2021.

[65] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[66] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron." https://github.com/facebookresearch/detectron, 2018.

[67] F. Rezaeianaran, R. Shetty, R. Aljundi, D. O. Reino, S. Zhang, and B. Schiele, "Seeking similarities over differences: Similarity-based domain alignment for adaptive object detection," Oct. 2021.

[68] M. Khodabandeh, A. Vahdat, M. Ranjbar, and W. G. Macready, "A robust learning approach to domain adaptive object detection," Apr. 2019.

[69] H.-K. Hsu, C.-H. Yao, Y.-H. Tsai, W.-C. Hung, H.-Y. Tseng, M. Singh, and M.-H. Yang, "Progressive domain adaptation for object detection," Oct. 2019.

[70] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," Mar. 2017.

[71] T. Kim, M. Jeong, S. Kim, S. Choi, and C. Kim, "Diversify and match: A domain adaptive representation learning paradigm for object detection," May 2019.

[72] Q. Cai, Y. Pan, C.-W. Ngo, X. Tian, L. Duan, and T. Yao, "Exploring object relation in mean teacher for cross-domain detection," Apr. 2019.

[73] Y.-C. Liu, C.-Y. Ma, Z. He, C.-W. Kuo, K. Chen, P. Zhang, B. Wu, Z. Kira, and P. Vajda, "Unbiased teacher for semi-supervised object detection," Feb. 2021.

[74] "Pytorch: Illustration of transforms," 2022. https://pytorch.org/vision/main/auto_examples/plot_transforms.html#sphx-glr-download-auto-examples-plot-transforms-py/, last accessed on 2022-07-11.

[75] M. Hnewa and H. Radha, "Multiscale domain adaptive yolo for cross-domain object detection," *IEEE International Conference on Image Processing (ICIP), 2021, pp. 3323-3327,*, June 2021.

[76] S. Zhang, H. Tuo, J. Hu, and Z. Jing, "Domain adaptive yolo for one-stage cross-domain detection," June 2021.

[77] C.-C. Hsu, Y.-H. Tsai, Y.-Y. Lin, and M.-H. Yang, "Every pixel matters: Center-aware feature alignment for domain adaptive object detector," 2020.

[78] V. Vidit and M. Salzmann, "Attention-based domain adaptation for single stage detectors," June 2021.

[79] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," Mar. 2014.

[80] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," Oct. 2013.

[81] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," June 2016.

[82] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," Aug. 2017.

[83] "Autodesk Navisworks API," 2022. https://apidocs.co/apps/navisworks/2018/87317537-2911-4c08-b492-6496c82b3ed0.htm/, last accessed on 2022-06-29.

[84] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation," *European Conference on Computer Vision (ECCV)*, 2018.

[85] T. Hodaň, F. Michel, E. Brachmann, W. Kehl, A. Glent Buch, D. Kraft, B. Drost, J. Vidal, S. Ihrke, X. Zabulis, C. Sahin, F. Manhardt, F. Tombari, T.-K. Kim, J. Matas, and C. Rother, "BOP: Benchmark for 6D object pose estimation dataset format," *European Conference on Computer Vision (ECCV)*, 2018. https://github.com/thodan/bop_toolkit/blob/master/docs/bop_datasets_format.md/, last accessed on 2022-06-29.

[86] N. Zeng, "An Introduction to Evaluation Metrics for Object Detection," 2022. https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/, last accessed on 2022-07-04.

[87] J. Jiang, B. Chen, J. Wang, and M. Long, "Decoupled adaptation for cross-domain object detection," *ICLR 2022*, Oct. 2021.

[88] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[89] C. Chen, Z. Zheng, X. Ding, Y. Huang, and Q. Dou, "Harmonizing transferability and discriminability for adapting object detectors," Mar. 2020.

[90] V. F. Arruda, T. M. Paixão, R. F. Berriel, A. F. D. Souza, C. Badue, N. Sebe, and T. Oliveira-Santos, "Cross-domain car detection using unsupervised image-to-image translation: From day to night," July 2019.

[91] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," Aug. 2016.

[92] C.-D. Xu, X.-R. Zhao, X. Jin, and X.-S. Wei, "Exploring categorical regularization for domain adaptive object detection," Mar. 2020.

[93] C. Peng, K. Zhao, and B. C. Lovell, "Faster ilod: Incremental learning for object detectors based on faster rcnn," *Pattern Recognition Letters*, vol. 140, pp. 109–115, 2020.

[94] A. B. Jung, K. Wada, J. Crall, S. Tanaka, J. Graving, C. Reinders, S. Yadav, J. Banerjee, G. Vecsei, A. Kraft, Z. Rui, J. Borovec, C. Vallentin, S. Zhydenko, K. Pfeiffer, B. Cook, I. Fernández, F.-M. De Rainville, C.-H. Weng, A. Ayala-Acevedo, R. Meudec, M. Laporte, *et al.*, "imgaug." https://github.com/aleju/imgaug, 2020. Online; accessed 01-Feb-2020.

A Appendices

A.1 Architecture

Listing 1: The proposed architecture of the added elements (pseudo-code)

```
## image-level domain adaptation
(RCNN_Image_DA):
(
     (conv1): Conv2d(1024, 256, kernel_size=(3, 3), stride
    =(1, 1), padding=(1, 1))
     (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=
    True, track_running_stats=True)
     (conv2): Conv2d(256, 128, kernel_size=(3, 3), stride=(1,
     1), padding=(1, 1))
     (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=
    True, track_running_stats=True)
     (conv3): Conv2d(128, 128, kernel_size=(3, 3), stride=(1,
     1), padding=(1, 1))
     (bn3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=
    True, track_running_stats=True)
     (classifier): Conv2d(128, 1, kernel_size=(3, 3), stride
    =(1, 1), padding=(1, 1))
     (leaky_relu): LeakyReLU(negative_slope=0.2, inplace=True
    )
)
## instance-level domain adaptation
(RCNN_Instance_DA):
(
     (fc_1): Linear(in_features=1024, out_features=256, bias=
    True)
     (leaky_relu_1): LeakyReLU(negative_slope=0.2, inplace=
    True)
     (dropout_1): Dropout(p=0.5, inplace=False)
     (fc_2): Linear(in_features=256, out_features=256, bias=
    True)
     (leaky_relu_2): LeakyReLU(negative_slope=0.2, inplace=
    True)
     (dropout_2): Dropout(p=0.5, inplace=False)
     (classifier): Linear(in_features=256, out_features=1,
    bias=True)
     (leaky_relu_3): LeakyReLU(negative_slope=0.2, inplace=
    True)
)
## consistency loss term between the two alignments
(consistency_loss): MSELoss()
```

```
27
28 ## module for continual learning
29 (FastRCNNExtendedOutputLayers):
30 (
31       ## for the original 20 classes and the background
32       (cls_score): Linear(in_features=1024, out_features=21,
     bias=True)
33       (bbox_pred): Linear(in_features=1024, out_features=80,
     bias=True)
34       (cls_score_extra_classes): Linear(in_features=1024,
    out_features=2, bias=True)
35        ## for one added class and the background
36       (bbox_pred_extra_classes): Linear(in_features=1024,
    out_features=4, bias=True)
37 )
```