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** 62+1 **Language** English

Abstract

TODO

fix this

Keywords Computer vision, object detection, transfer learning, domain adaptation, 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	Regularization 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

verify this

$\frac{\mathrm{d}}{\mathrm{d}t}$	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

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
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
FPN	Feature Pyramid Networks
YOLO	You Only Look Once
SSD	Single-shot MultiBox detector
IoU	Intersection over Union
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
mAP	mean Average Precision
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

Todo list

dont forget to disable todos

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]. However, although computer vision has been integrated into industrial applications (e.g., safety and process monitoring)[4][1], less research has addressed the issue of equipment detection.

As industrial plants are typically hundreds of meters long, it often becomes frustrating to identify 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 [5], for detecting defects in copper molds[6] , as well as for recognizing froth in flotation cells[7], 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 an application that would further optimize the processes by leveraging state-of-the-art computer vision algorithms in equipment recognition.

Add missing citations

Even though various methods have been implemented for detection of objects in a countless number of fields[8], 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 gold 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[9] . The domain shift occurs when the environmental conditions at the time of capturing training and test images change, as discussed in [9].

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 equipment 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 incrementally 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.

1.3 Methodology

Refine after all done

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 used to train a 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 based on the T-LESS open-source dataset [10]. 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 [11] algorithm for cross-domain object detection, which in turn uses the Faster-RCNN [12] implementation in the detectron2 [13] framework as a detector base. The thesis will contribute to current knowledge by introducing an instance-level domain classifier appended to the base network of the Adaptive Teacher algorithm. Additionally, the study will evaluate the feasibility of other strategies, such as continual learning [14], 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 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 training.

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 in production. 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 literature on object detection models, domain adaptation, the latest cross-domain object detection techniques, and class incremental learning implementations. Chapter 3 defines the dataset used and outlines the proposed architecture of the model. Chapter 4 evaluates the solution and compares the results to those of other methods using average precision metrics. 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, trasnfer learning and domain adaptation. Additionally, the section will familiarize the reader with the relevant terminology and notations used. 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 subfield of Artificial Intelligence(AI), has been a highly computational task. The primary cause of it was linked to low hardware performance. According to the Moore's law [15], in a given number of months, the amount of transistors doubles in a circuit. As the computational power of the computers grew proportianally 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.[16] Few additional critical bottlenecks in ML are caused by suboptimal 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).

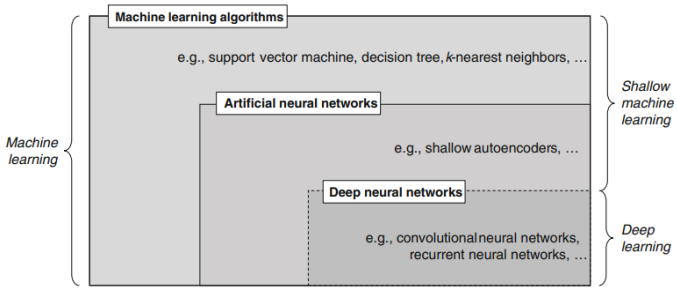


Figure 1: Machine learning concepts [17]

The concept of 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 cellbody 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 on Figure 2(c). Identical process takes place in remaining neurons, which ultimately forms a neural network.[18]

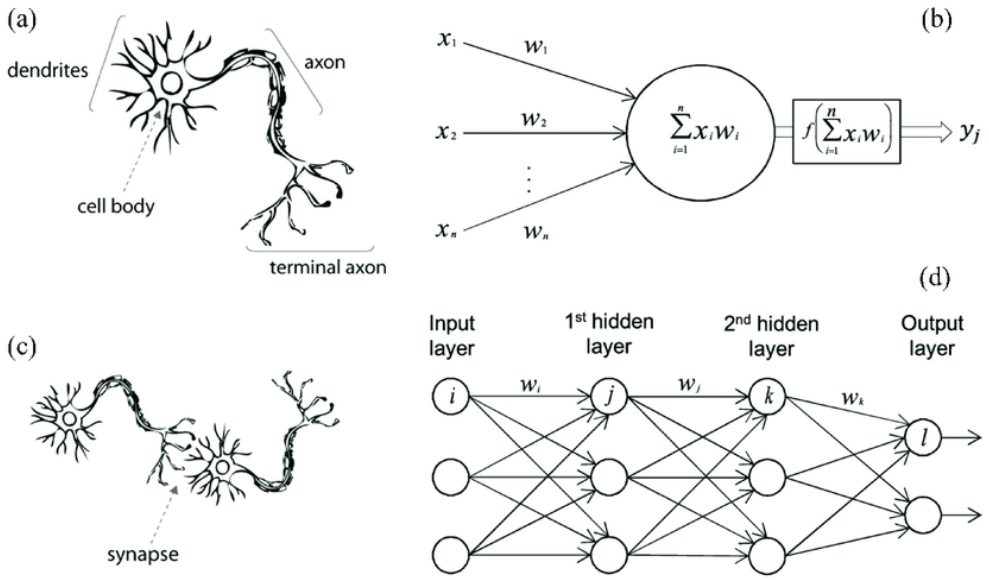


Figure 2: A biological(a) neuron against artificial(b) and biological synapse(c) against artificial(d)[19]

In deep learning(DL), a subfield 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 $w_1, w_2, ...w_n$ are the weights of the signal. 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 allows to learn complex patterns in data. [18]

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) [20] 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 on 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. [21] 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 on 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[22]. A naive backpropogation approach is illustrated on the 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 in each layer of the neural network and returns a predicted output. In order to evaluate the quality of the predicted output, it is compared to the target output by using a cost(loss) function. The classic example of a cost function is a mean-squared error (MSE), which is commonly used in regression based problems:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} \left(f_\theta\left(x_i\right) - y_i\right)^2 \tag{2}$$

The MSE cost function attempts to minimize the distance between the predicted output and the target output, while giving more weight to larger distances due to the squared output. [23]

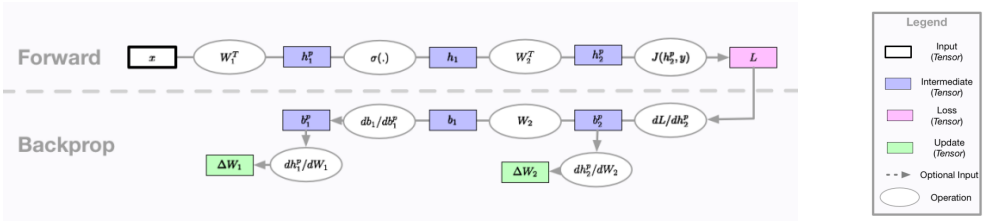


Figure 3: Backpropogation algorithm, adapted from [24]

In order to minimize the loss function, the algorithm calculates the partial derivative of the loss term L with respect to the weights: $\frac{\partial L(f(x),y)}{\partial W_i}$. An algorithm of gradient descent is commonly used in optimizing such functions and its logic can be generalized as follows:

1. Start with $j = 0$ and a random value of θ, called θ^0.
2. Set θ^{j+1} to $\theta^j - \eta\left(\left(\nabla g\right)\left(\theta^i\right)\right)$.
3. j to $j + 1$ and repeat.[23]

Make sure this formula makes sense with the rest of the text

Generally speaking, there are three gradient descent algorithms: 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 at the time. On the other hand, the mini-batch gradient descent algorithm splits 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 ΔW are calculated for each of weights the in the layers. [24] The algorithm is then repeated until convergence. 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".

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. [25] Nowadays, CNNs are typically used to tackle various pattern recognition and computer vision tasks. Some of the tasks include:

- Image Classification
- Object Detection
- Segmentation
- Facial Recognition and Modelling
- Domain Adaptation
- Image Reconstruction
- along with many others [26]

For addressing the objectives of this work, the thesis will extensively cover image classification, object detection and domain adaptation tasks.

Figure 4 illustrates a simplistic CNN architecture approach to the MNIST [27] 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 appropriate algorithms for learning. On the countrary, DL algorithms, such as CNN, learn the features automatically. [28]

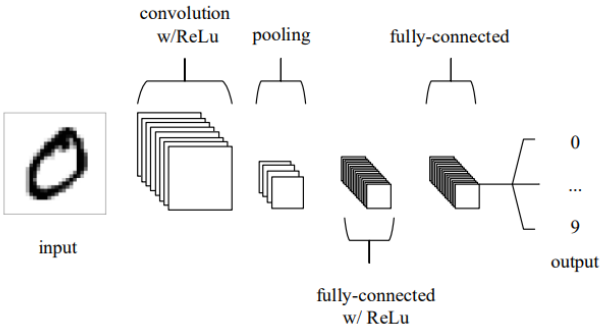


Figure 4: A simple CNN network with 5 layers for image classification task [25]

To understand the logic of 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 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.[29] Figure 5(a) illustrates the process of convolution in image processing.

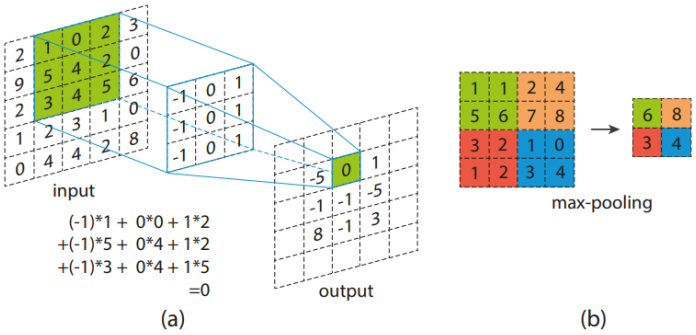


Figure 5: The process of convolution [29]

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 downsample the image and hence reduce the computational costs. [29] The commonly used max-pooling operation is shown in Figure 5(b). Finally, fully-connected layers complete the structure of the basic CNN. FC layers are used to flatten the ouputs from the convolutional layers. This in turn allows, in case of the input in Figure 4, to compute the probability of the input image to represent a number. [25]

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. The datasets were made public as different image classification challenges appeared. The datasets commonly used in benchmarking are ImageNet[30], PASCAL Visual Object Classes(VOC)[31] and Common Objects in Context(COCO) [32].

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[33], AlexNet[34], VGGNet[35] and Residual Neural Network(ResNet) [36]. As it can be concluded from the Figure 6, the classification error dropped lower than the error from the human eye with the introduction of ResNet, thus approaching the theoretical limits.

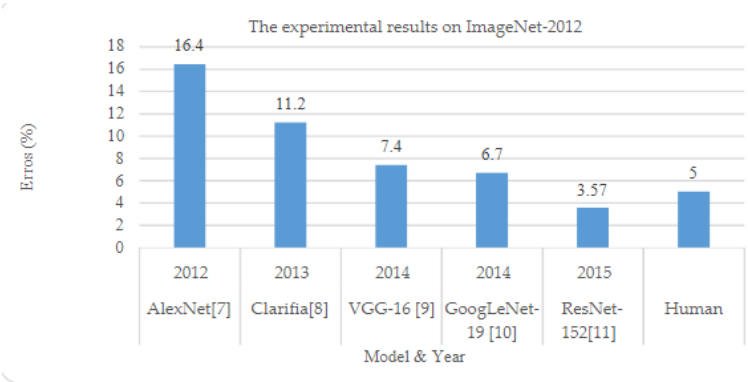


Figure 6: Evolution of image classifier models evaluated on ImageNet dataset [28]

2.3.1 LeNet

LeNet architecture (1998)[33] is considered to be a pioneer in the field. Its design inherited the classic CNN architecture, although instead it consisted of 7 layers, as presented on the Figure 7. However, the implementation of the paper was not possible for more than 10 years due to limitations in computing power.

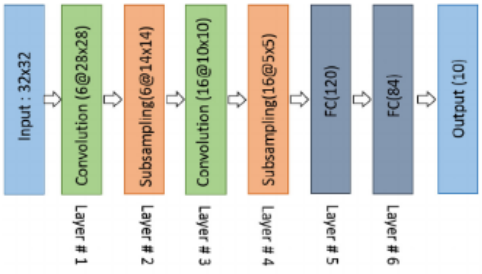


Figure 7: LeNet architecture [28]

2.3.2 AlexNet

Consequently, AlexNet paper was introduced, which proved the effectiveness of their model, as it outperformed the state-of-the-art implementations and achieved the error rate of 15.3%. [34] 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 followed byt ReLU activation functions and max-pooling layers. The architecture is finalized by attaching two FC layers with dropout rate of 0.5 and one 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 overfitting in FC layers and improve generalization. [37]

Overall, 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.

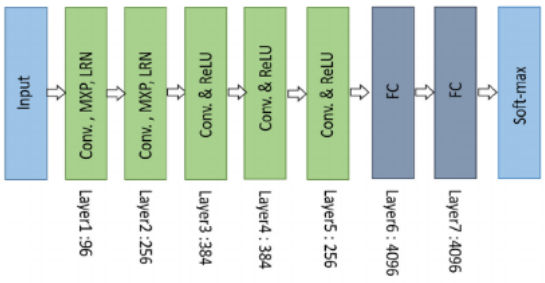


Figure 8: AlexNet architecture [28]

2.3.3 VGG

Another milestone was achieved with the discovery proposed in VGGNet[35]. This work proved that the depth of the network produces a significant impact on the performance of the CNN in classification tasks.[28] 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 finilized with three FC and one softmax layers.

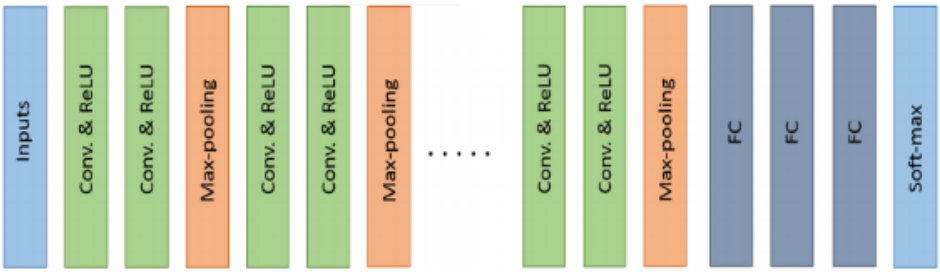


Figure 9: VGG architecture [28]

2.3.4 ResNet

ResNet architecture [36],which was proposed in 2015, discovered that after certain depth, the performance of the model degrades. The authors 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. The authors proposed to utilize a residual block, illustrated on 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 affect the performance negatively.

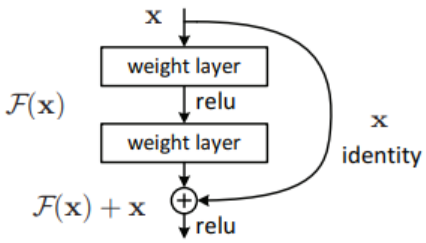


Figure 10: Residual block of ResNet[36]

ResNet adopts the VGG-19 architecture, with an exception of the skip connection block added. The authors implemented multiple versions of the model, including one that is 152 layers deep. Nonetheless, the network has less trainable parameters, 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 error. [36]

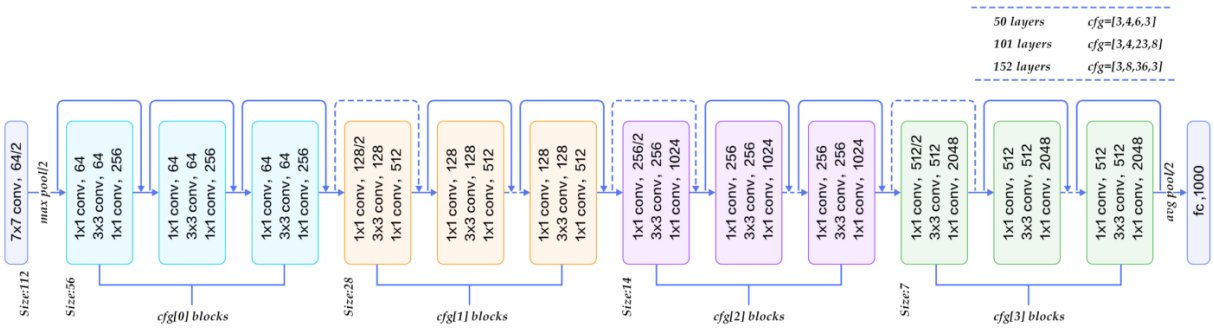


Figure 11: ResNet architecture[38]

2.4 Object detection

The problem of 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 pretrained VGG or ResNet models, presented in the previous chapter. A base network acts as a feature extractor, and the features are then passed to a detector. Generally speaking, deep learning based detector networks can be classified into two categories: single- and two-stage detectors. [39] Two-stage object detectors attempt to propose regions of the image that contain an object in the first step, and then run the task of classification and localization on the given proposal region. On the other hand, single-stage detectors try to detect objects directly without running a Region Proposal Network (RPN). Figure 12 illustrates the key differences in the structure of the two categories.

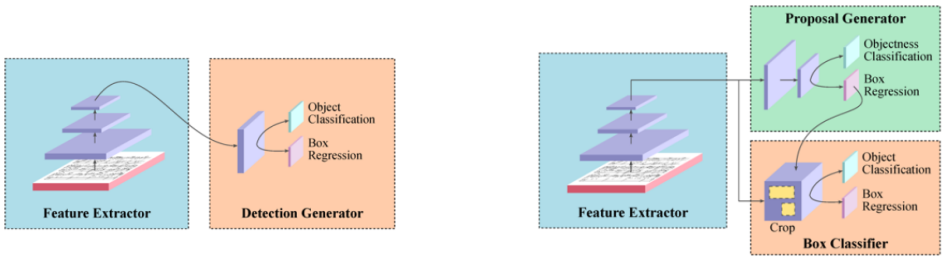


Figure 12: A simple single-stage detector(left) compared to a two-stage detector(right) [40]

Typical examples of the single-stage networks are RetinaNet[41], Single-shot detector(SSD)[42] and You only look once(YOLO)[43]. In two-stage detectors, key contributions were made by the authors of the models such as the Region-based CNN(R-CNN)[44], Fast-RCNN[45] and Faster-RCNN[12]. Figure 13 also mentions detectors such as FCOS[46], Mask-RCNN[47] and DETR[48]. 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.

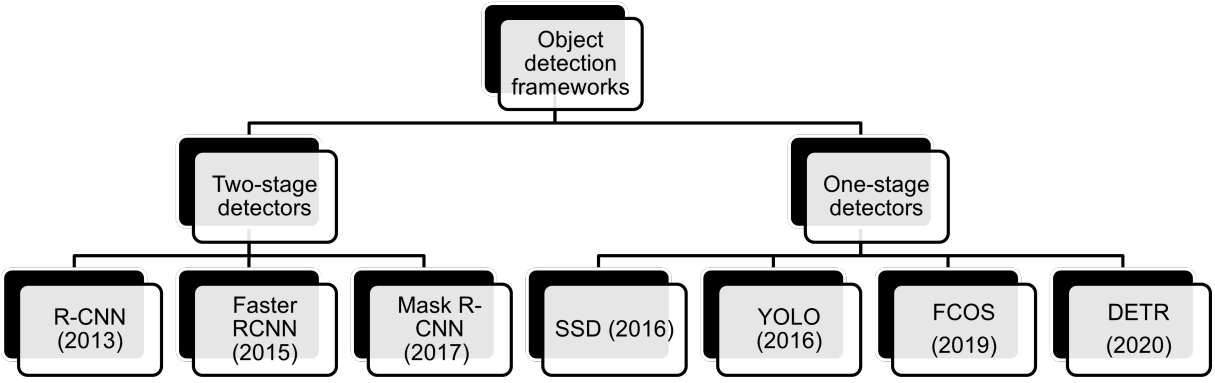


Figure 13: Types of object detectors

2.4.1 R-CNN

Figure 14 shows the structure of the R-CNN network[44]. R-CNN has been one of the first CNN-based models to be introduced [39]. The authors essentially suggested to use an 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 [49] was used to produce roughly 2000 regions. AlexNet [34] was used as a backbone CNN to extract vectors of 4096 size dimensions. The features were then passed to binary classifers that are bound to a certain class. As multiple regions are returned, non-maxumum suppression(NMS)[50] is used to identify the best proposal for the object.

Unfortunately, the inference process in R-CNN took a whole 47 seconds per image[44], which was not fast enough to be remain relevant in object detection field.

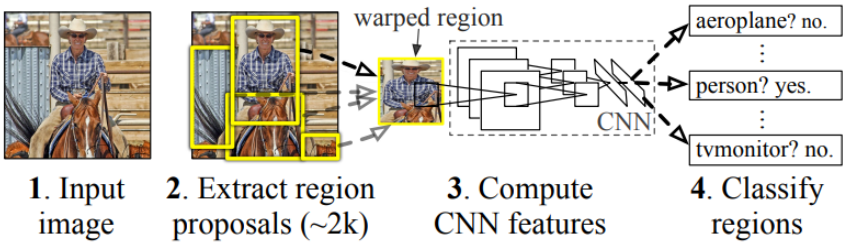


Figure 14: R-CNN overview[44]

2.4.2 Fast-RCNN

Write more about ROI

Unlike the classic R-CNN system, where the components such as the classifier and the regressor have to be trained separately, the authors of the Fast-RCNN [45] paper introduced an end-to-end trainable system. Additionally, the region-of-interest(ROI) pooling layer was proposed. The architecture of the Fast-RCNN shown in Figure 15 below. The ROI layer receives max-pooled feature maps that were generated by the backbone CNN and then for every proposed region, a fixed size feature vector is extracted from the map. 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 one of the object classes K in range (0, K+1], thus including the background class as well. The other one returns 4 real values for each feature vector, which denote the boudning box corner coordinates excluding the background class. [45] Fast-RCNN showed a significant(x146) speed improvement as compared to the R-CNN, thus allowing it to be used in real-time applications[44].

Write more about image pyramids

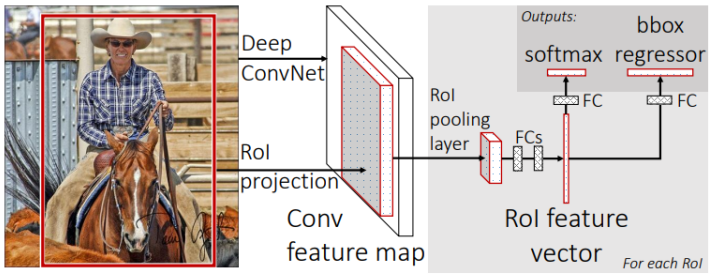


Figure 15: Fast-RCNN overview[45]

2.4.3 Faster-RCNN

Perhaps write in greater detail

The authors of [12] discovered that even though Fast-RCNN was significantly faster than the precessor networks, this architecture still had a bottleneck remaining in its region proposal counterpart. Therefore, in [12] it was suggested to replace the legacy region proposal module with a fully-convoluted architecture called region proposal network(RPN)[44]. Instead of using image pyramids to solve the problem as it was proposed in [45], Faster-RCNN [12] 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 on 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 proposed object's class. [12]

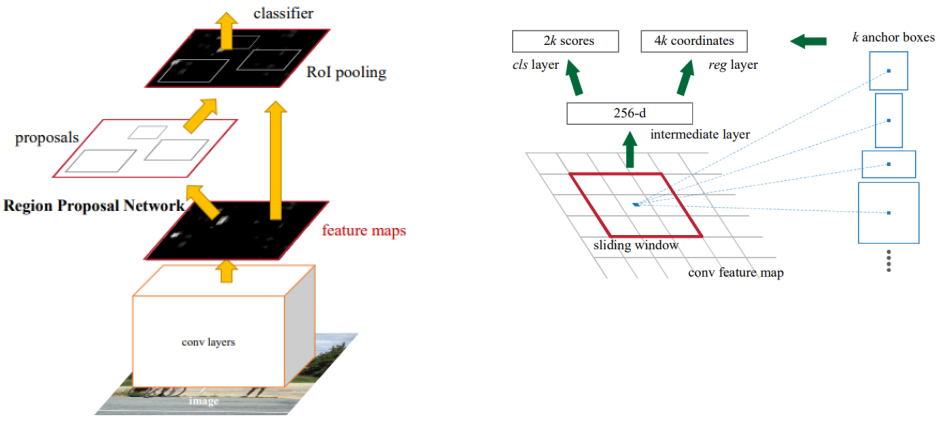


Figure 16: Faster-RCNN overview and its RPN module [12]

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 at the time of writing this thesis, it is still one of the most widely used detectors in the field, as can be noticed from the 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 real-timeness as compared even to the fastest one of the two-stage detectors, Faster-RCNN.

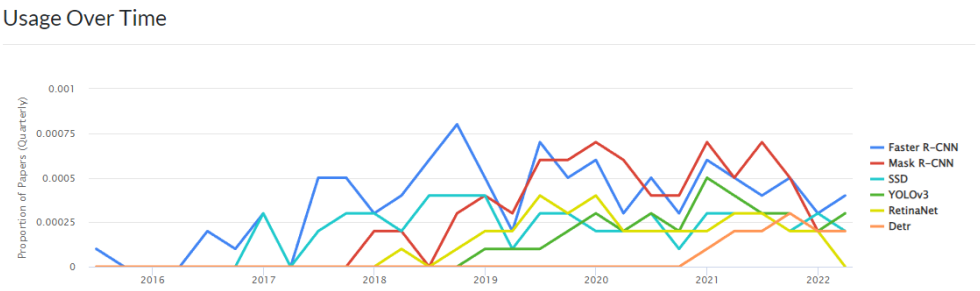


Figure 17: The popularity of different detectors according to [51]

2.4.4 YOLO

Unlike the two-stage methods presented, You Only Look Once(YOLO)[43] algorithm attempts to solve the object detection problem purely as a 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 boundning boxes and C probabilities of the class and are predicted. [43] 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[44], where the algorithm of selective search [49] 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. The authors reported YOLO to sustain the average FPS of 45, while the most accurate version of the Faster-RCNN network had 7 FPS [43]. However, the proposed network still has multiple limitations, which were addressed in later iterations of the paper [52][53].

The architecture consists of 24 cascaded convolutional and 2 FC layers.The convolutional layers are first pretrained on the ImageNet dataset[30]. The image size is reduced by applying 1×1 convolutional layers in between, also referred to as "reduction layers". [43] The simplified architecture is presented in Figure 19.

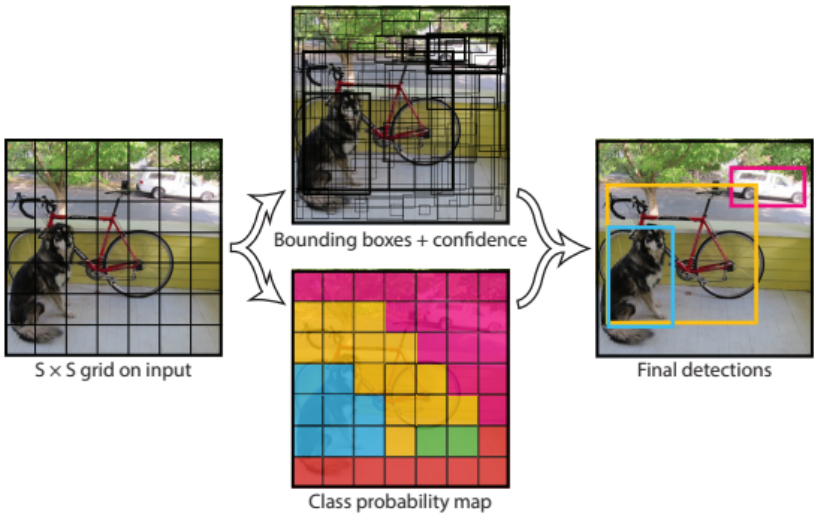


Figure 18: YOLO overview[43]

2.4.5 SSD

Another implementation of a single-stage network worth mentioning is a Single-Shot MultiBox detector(SSD)[42]. The differences in architectures of SSD and YOLO

are displayed on the Figure 19. The authors of SSD 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 benetits from the anchor boxes of varying aspect ratio.

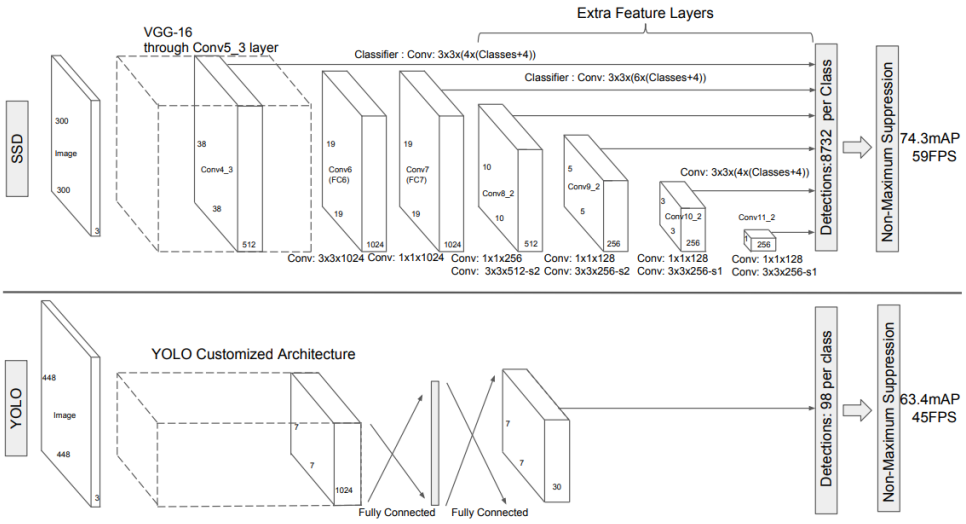
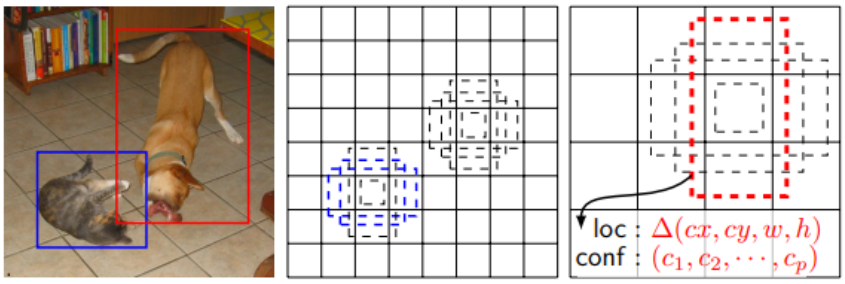


Figure 19: SSD compared to YOLO[42]

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 an offset. The anchor boxes with offsets that have the highest overlap with the ground truth box of the object are then oassed to FC layers to predict the class and the location of the object.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 20: SSD anchor boxes[42]

The key object detection models of the last decade are presented [39] in the Table 1 below. As it can be deducted from the table, Faster-RCNN shows the best performance on the PASCAL VOC dataset, while YOLO demonstrated the highest FPS. The survey [39] 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.

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).

Table 1: Overview of object detectors[39]

As it can be seen from the Table 1, there are multiple models that are not covered in this section. Additionally, there are several detectors that show promising results in the domain-adaptive object detection setup, such as FCOS [46] and DETR [54]. However, as their performance is not yet extensively evaluated, they will be omitted from this thesis.

2.5 Transfer learning

In this section, the multiple transfer learning(TL) techniques will be introduced. Despite the fact that the methods discussed in 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 time- and money-wise, and sometimes simply not possible at all [55]. 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 it addresses transferring knowledge learned 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. [56] and are introduced as follows:

mention later: Zhuang2019: TL does not always bring benefit

- A domain $\mathcal{D}$ can be defined as a composite term, which is characterized by two elements: feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$; $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$. With this being said, two domains are defined as different if their feature spaces or marginal probability distributions are different. In this thesis, the domains are restricted with one source $\mathcal{D}_S$ and one target $\mathcal{D}_T$ domains [56].
- Similarly, a task $\mathcal{T}$ can be defined by its label space $\mathcal{Y}$ and a 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 [56].

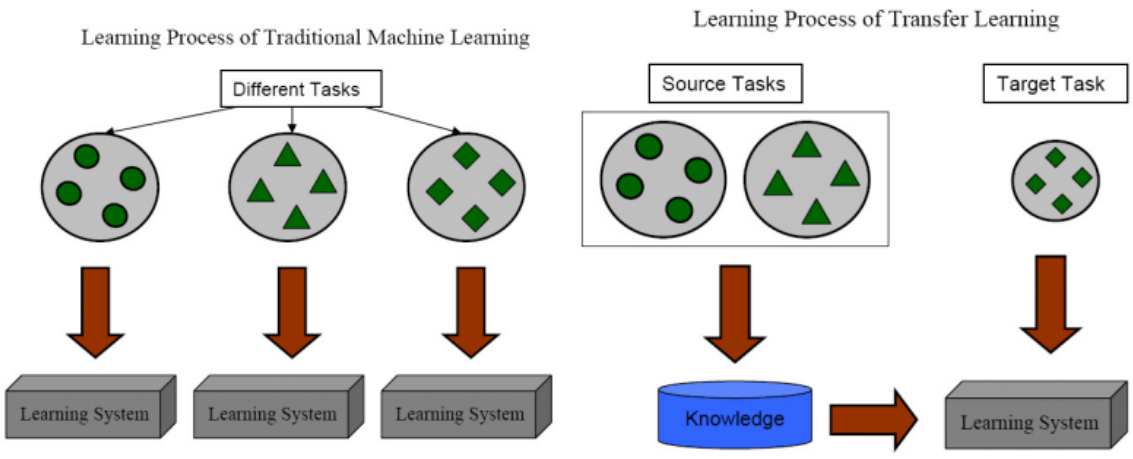


Figure 21: Comparison of ML to TL[56]

Sun et al. [57] describes TL as a method to transfer knowledge when the feature space between the two given source and target domains is different, meaning that $\mathcal{X}_S \neq \mathcal{X}_T$. Pan et al. [56] 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 [58] and [59], domain adaptation(DA) is a special case of TL. In scientific terms, the problem that DA attempts to solve can be defined when the source and target feature spaces are the same $\mathcal{X}_S = \mathcal{X}_T$, but the marginal probability distribution is not $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$[57]. This is not to be confused with semi-supervised machine learning, where both labeled and unlabeled data is typically supplied from one domain[57].

Talk about the comment here perhaps

2.5.1 Domain adaptation

DA has been gaining popularity lately[59] in both image classification and object detection tasks. Typically, DA is used to predict a label given the data from a source domain and limited or no data from the target domain. Most importantly, DA addresses the domain shift problem [59]. 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. [59] classify DA methods into three categories that are similar to ML types - supervised, semi-supervised, and unsupervised DA. Alternatively, Oza et al. [60] 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 shift when the data availability is limited.

According to Zhang et al.[59], different methods that attempt to solve the distribution shift problem by minimizing the distance between marginal, conditional or joint distributions.

The visual representation of these distribution alignment types are shown on Figure 22 and the methods in the following subsections attempt to minimize the domain shift by one way or another.

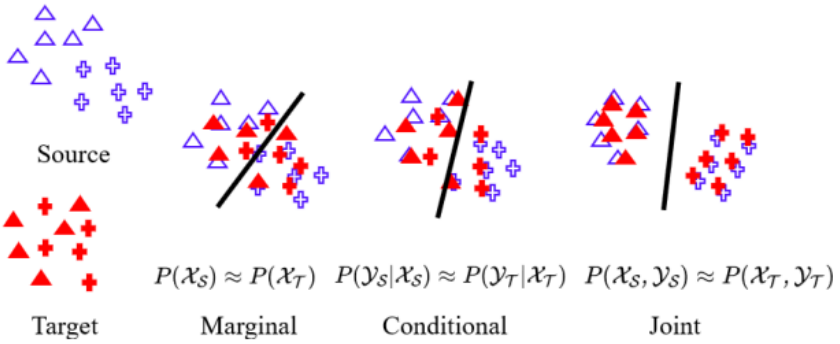


Figure 22: Distribution alignment types[59]

2.6 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. [60] 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 on Figure 23.

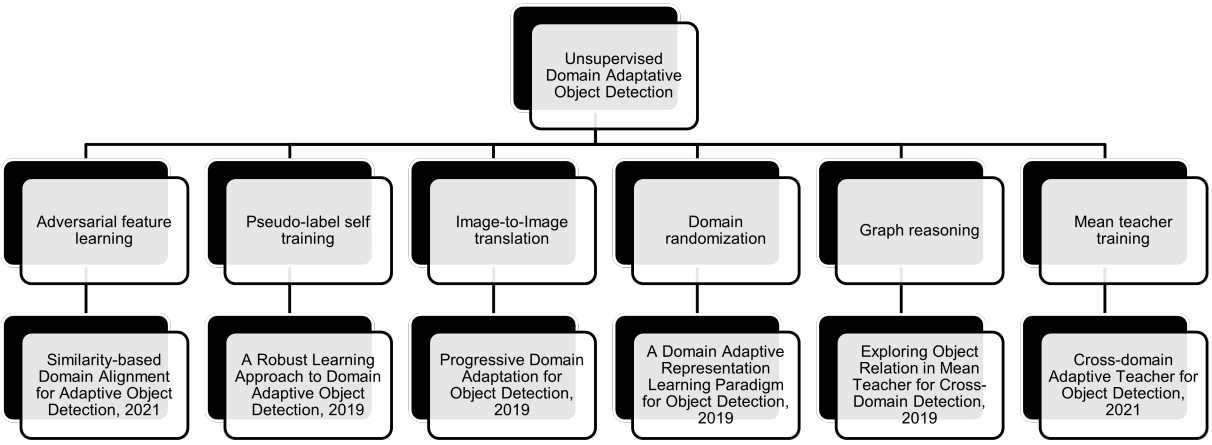


Figure 23: Unsupervised Domain Adaptative Object Detection

In the following methods, it will be assumed that $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$ originate from similar yet different distributions. Additionally, the papers introduced in the following section mainly address the more complex, unsupervised domain adaptation(UDA) problem.

2.6.1 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 proposed by Ganin et al[9]. The authors suggest that in order to successfully solve the domain adaptation problem, the prediction should be based on the features that cannot differentiate between the source and target domains. In other words, the network should propose features that are common for both domains. The Figure 24 illustrates the Domain-Adversarial Neural Network, or DANN[9].

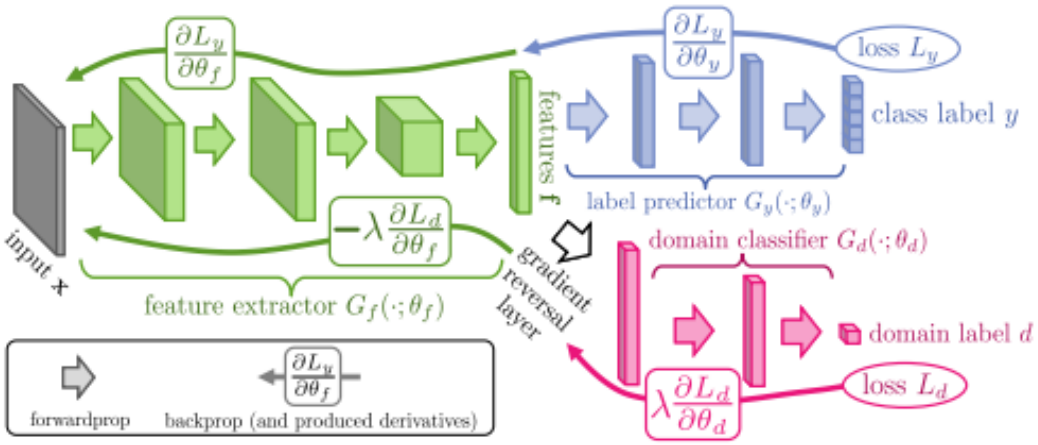


Figure 24: Domain-adversarial neural network and GRL[9]

The network is essentially a simple feed-forward network with a feature extractor and a predictor that classifies the input. Two additional components are appended to the last layer of the feature extractor - a gradient reversal layer(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 acts as an identity function. 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 maximize the domain classification loss and, therefore, "confuse" the feature extractor and force it to generate only domain invariant features. [9]

Perhaps add more math about GRL

2.6.2 Adversarial feature learning

Although DANN is an implementation of a domain adaptive image classification, GRL is a fundamental component in adversarial feature learning of object detectors and will be referenced in the subsequent sections of the thesis.

The majority of the methods, collected by Oza et al.[60], are based on the two-stage object detectors and Faster-RCNN [12] in particular. This has been presumably facilitated by Pytorch [61] package in Python and frameworks such as Detectron [62] and Detectron2 [13] 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[63]. The proposed architecture is based on the Faster-RCNN object detector. As it can be seen from Figure 25, this method proposed to

Write more about this as it is key of your method

apply adversarial learning at multiple stages of the detection, namely in the image- and instance-level of the network. To be more precise, the GRL and the domain classifier are appended to the extracted feature map, same way as in Figure 24 to form an image-level domain classifier. Similarly, features extracted from the ROI follow the same procedure. Finally, the classifiers are regularized using consistency loss. The objective of the network is then defined as $\max_{D_{inst},D_{img}} \min_F \mathcal{L}_{\text{det}}^{frcnn} - \lambda\left(\mathcal{L}_{img} + \mathcal{L}_{inst}\right)$, meaning that the network attempts to minimize the detection loss, while maximizing the image-level loss $\mathcal{L}_{img}$ and instance-level loss $\mathcal{L}_{inst}$ and minimize the consistency loss $\lambda\mathcal{L}_{\text{consistency}}$. [63].

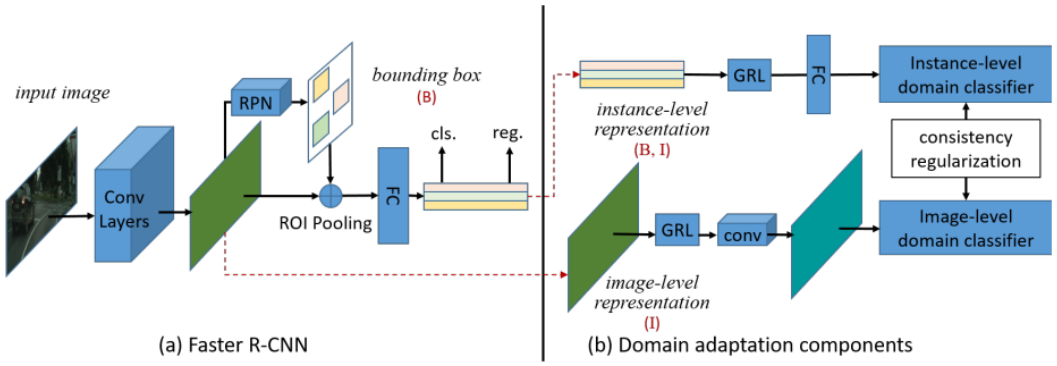


Figure 25: Domain Adaptive Faster R-CNN for Object Detection in the Wild[63]

Another most recent state-of-the-art study made by Rezaeianaran et al. [64] proposed different approach that compares the adversarial training to contrastive learning. Similarly to [63], the network leverages a Faster-RCNN detector with instance- and image-level losses. The key difference is in the way the latter two losses are calculated. Rezaeianaran et al. attempted to push the features closer if they represent the same class and push them apart otherwise by utilizing max-margin contrastive loss. The margin here denotes how far the features can be in order to be considered the same class. Constrastive loss took the form of Equation 3:

$$\mathcal{L}_{CL} = \sum_{i}^{C} \left[\left\| F_S^i - F_T^i \right\|_2^2 + \sum_{j,j\neq i}^{C} \max\left\{0, m - \left\| F_S^i - F_T^j \right\|_2^2\right\} \right] \text{[64]} \tag{3}$$

Additionally, as the the paper tried to solve UDA, no labels were available from the target domain and, therefore, pseudo-labelling was used to calculate the contrastive loss.

This paper review should be rewritten

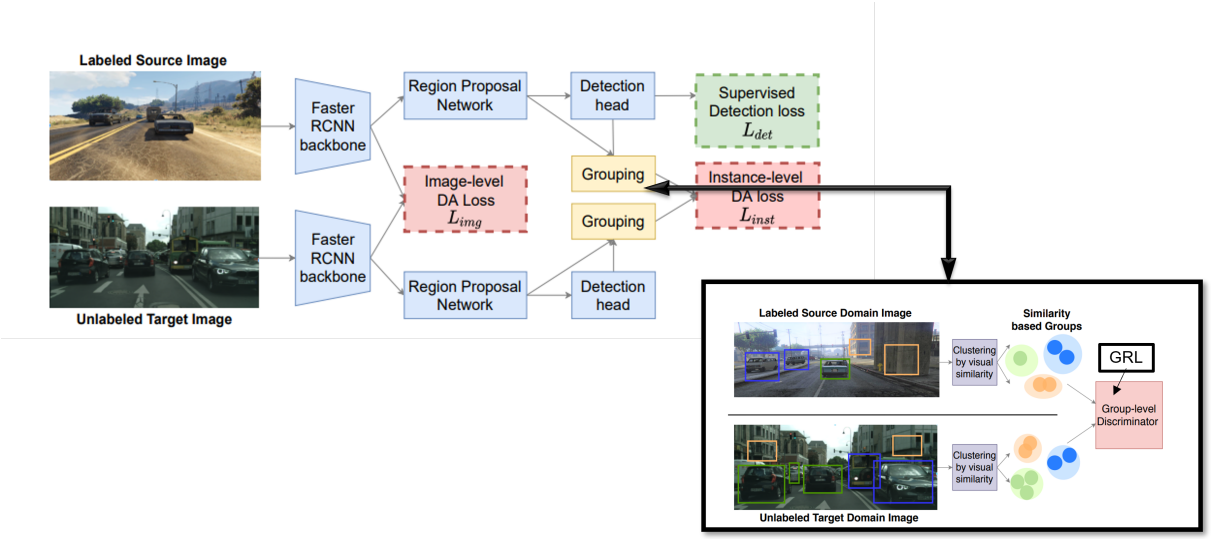


Figure 26: Seeking Similarities over Differences: Similarity-based Domain Alignment for Adaptive Object Detection, adapted from [64]

2.6.3 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-labelling is to first train the source dataset $\mathcal{D}_{\mathcal{S}} = \{\mathcal{X}_{\mathcal{S}}^i, \mathcal{Y}_{\mathcal{S}}^i\}_{i=1}^{\mathcal{N}_{\mathcal{S}}}$ and later run inference to obtain pseudo-labels on 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 $\dot{\mathcal{D}}_{\mathcal{T}} = \left\{\mathcal{X}_{\mathcal{T}}^j, \mathcal{Y}_{\mathcal{T}}^j\right\}_{j=1}^{\mathcal{N}_{\mathcal{T}}}$ [60]. 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.

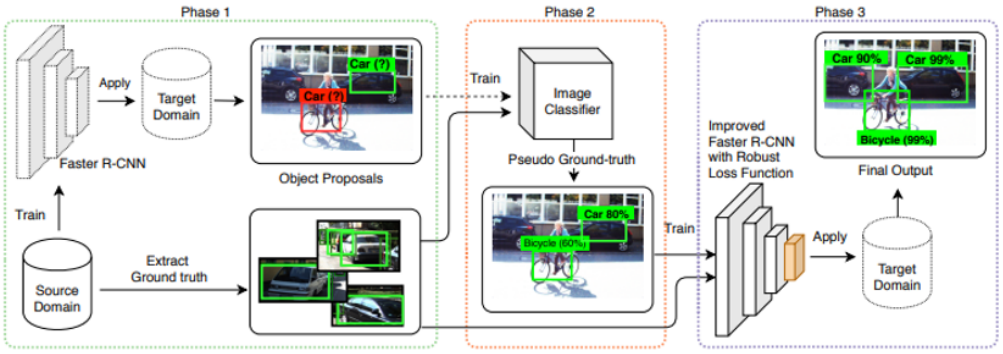


Figure 27: A Robust Learning Approach to Domain Adaptive Object Detection[65]

In the first stage, the network is treated as a typical 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 fed into a pretrained 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:

$$\min_{q} \mathrm{KL}\left(q\left(y_c\right) \| p_{cls}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right) + \alpha\, \mathrm{KL}\left(q\left(y_c\right) \| p_{img}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right) [65], \tag{4}$$

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. The goal of the refining process is to find a distribution $q(y_c)$ that is close to the models of p_{cls} and p_{img}. The process is relatively similar for the bounding box refinement and the reader is advised to consult the original paper for more details [65].

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 $\dot{\mathcal{D}}_{\mathcal{T}}$.

2.6.4 Image-to-Image translation

Another category that Oza et al.[60] outlined is image-to-image translation for UDA. Instead of trying to align features, this group of methods essentially attempt to pull the domains together first. Hsu et al.[66] has suggested a Cycle-Generative Adversarial Network(GAN)[67] based approach to transform images from the target domain into the source domain alike images. Figure 28 represents the complete network proposed by Hsu et al.

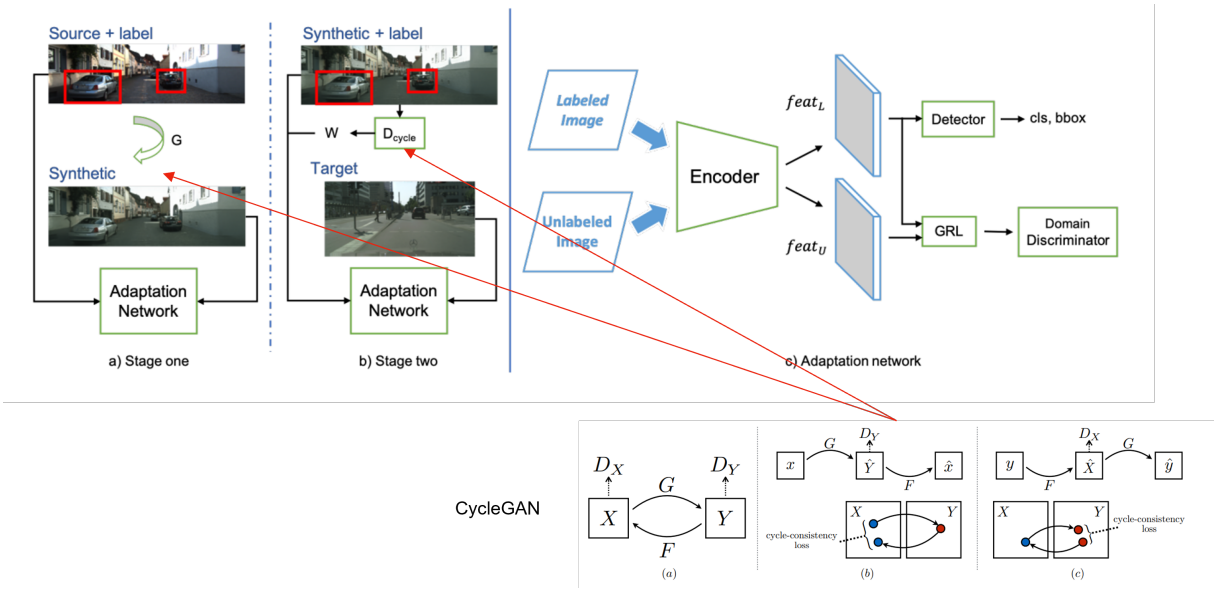


Figure 28: Progressive Domain Adaptation for Object Detection and CycleGAN, adapted from [66]

The generator part of the Cycle-GAN in Figure 28(a) creates intermediate domain from images in the source domain. The source images are then passed together with the labels to the Faster-RCNN network. The network is then tries to adapt the source domain to the sythetic domain. To further minimize the domain shift, adversarial learning techniques are used, such as a combination of a GRL and a domain classifier [66].

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 the features with the target domain. In the second stage of the training process, Hsu et al. proposed to utilize the weights w from the discriminator of the Cycle-GAN, which is additionally trained to differentiate between source and target domains. Ultimately, such approach allowed to amplify the importance of the synthetic samples that are closer to the target domain. The weighted loss from the discriminator was then summed with the detection loss and the adversarial loss to finally adapt the synthetic domain to target [66].

2.6.5 Domain randomization

Oza et al. [60] 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. [68]. 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. proposed a detector network that is based on Faster-RCNN with two additional components. The first component is a domain diversification module. Similarly to the method proposed by Hsu et al.[66], the module leverages a Cycle-GAN[67]. However, the diversification module is used to generate images in a fixed set of new domains rather than to match the source dataset with 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.

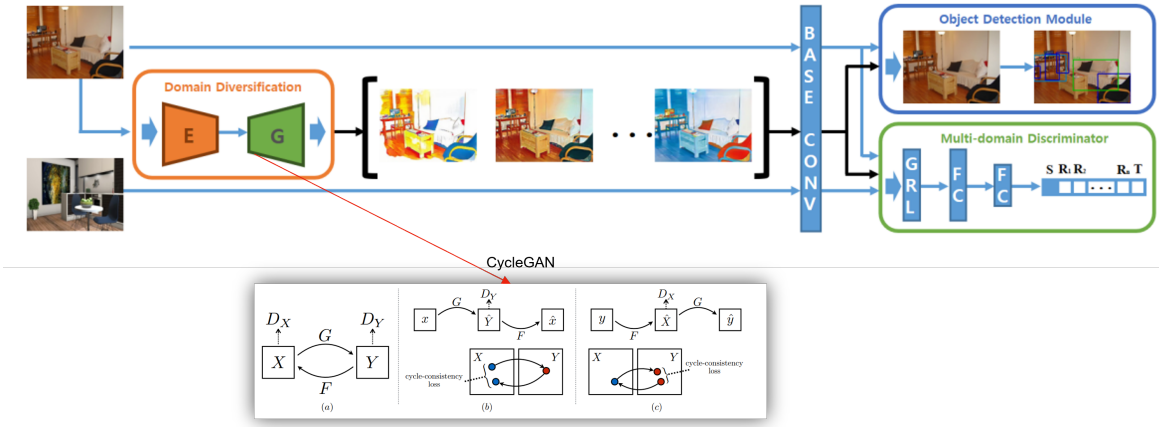


Figure 29: Diversify and Match: A Domain Adaptive Representation Learning Paradigm for Object Detection, adapted from [68]

During Cycle-GAN image generation, two additional loss terms limited 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 affect the model negatively [60]. In the original experiments the authors 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-invarian network with help of a multi-class domain classifier and the resulted model was used to verify performance on the target dataset.

2.6.6 Mean Teacher and 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 [60]. Cai et al. [69] proposed an architecture that combines both mean teacher and graph reasoning techniques in one solution and such architecture is presented in Figure 30.

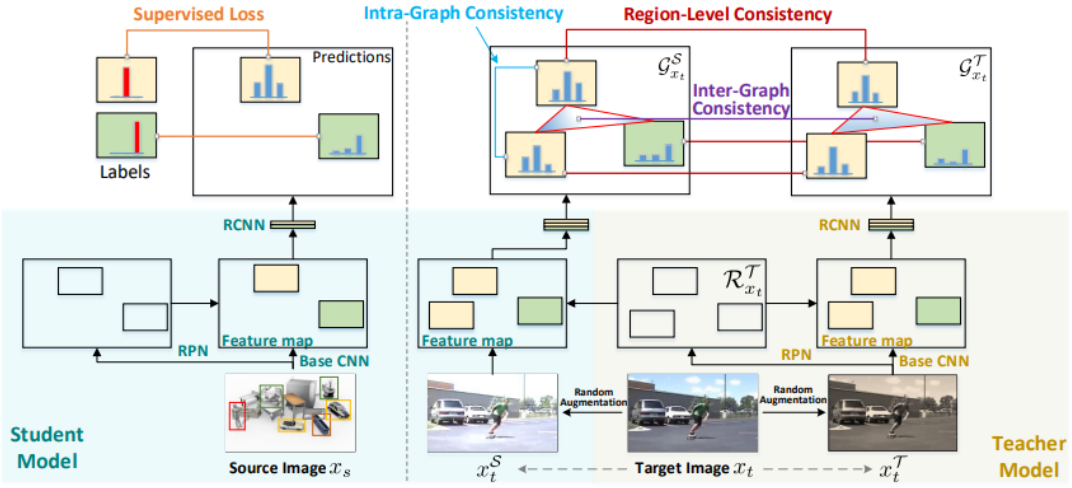


Figure 30: Exploring Object Relation in Mean Teacher for Cross-Domain Detection[69]

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 as a graph, each pixel can be considered a node, which is linked to all of its neighbours. The setup developed by Cai et al. [69] introduces a student-teacher framework based on Faster-RCNN that verifies the consistency 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. Images from the source domain are trained in a locked student environment in a supervised manner. Images from the target model, on the other hand, are augmented with in two iterations. First, the images are randomly cropped, padded or flipped. These images are passed through the pretrained supervised student model that generates predictions. Meanwhile, the original target images are augmented with color jittering or PCA noise, and this set is send to the teacher model. The teacher model also produces predictions, which are then compared with the set of predictions from the first round of augmented images by utilizing region-level consistency. Essentially, the region-level consistency is calculated as MSE of the region-level prediction error of both the student and the teacher.

Inter-graph level consistency is used to verify the quality of the two graphs produced by the teacher and student models. It is calculated by means of cosine similarity between the graph representations of the proposed regions.

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 UDA setup, $argmax(labels)$ 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 more detailed calculation of the loss terms the readers are referred to the original paper by Cai et al. [69]

A purely mean-teacher-based semi-supervised approach has been proposed by Liu et al.[70] As it can be seen from the overview illustrated in Figure 31, it consists of two sequential stages. During the burn-in stage, Faster-RCNN detector is trained on the labeled data as normal. Next, the training pipeline is split into two equivalent Faster-RCNN-based detectors. The teacher model is supplied with weakly-augmented data. On the other hand, strongly-augmented images are fed into the student model. According to Liu et al., the main reason for that was because while strong augmentation is needed to improve performance of the model, the weaker augmentations were still needed in the teacher model to generate reliable pseudo-labels. These pseudo-labels are in turn used in the student model.

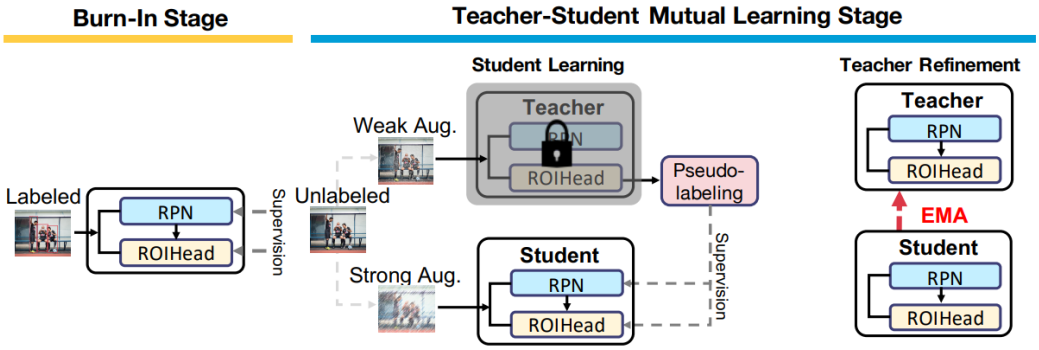


Figure 31: Unbiased Teacher for Semi-Supervised Object Detection[70]

In order to achieve higher accuracy of pseudo-labels, the unbiased teacher model includes an Exponential Moving Average(EMA) module, along with other techniques [70]. EMA attempts to emphasize the most recent data by granting it higher weight. The EMA in Unbiased teacher is defined as follows:

$$\theta_t^i = \hat{\theta} - \gamma \sum_{j=1}^{i-1} \left(1 - \alpha^{-j+(i-1)}\right) \frac{\partial \left(\mathcal{L}_{\text{sup}} + \boldsymbol{\lambda}_u \mathcal{L}_{unsup}\right)}{\partial \theta_s^j} [70], \tag{5}$$

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.

The weight of EMA in the training process is defined by α and γ is the learning rate of the ensembled model [70].

Liu et al. also discusses the class imbalance problem that often causes the detector to learn underrepresented classes poorly [70]. In order to solve this problem, the authors propose to make use of the multi-focal loss [41], 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.[11] proposed to utilize mean teacher training in a domain adaptation setup. In addition to the original ensembled network proposed by Liu et al. [70], adaptive teacher architecture employs adversarial feature learning techniques to adapt the target domain to the source. 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.

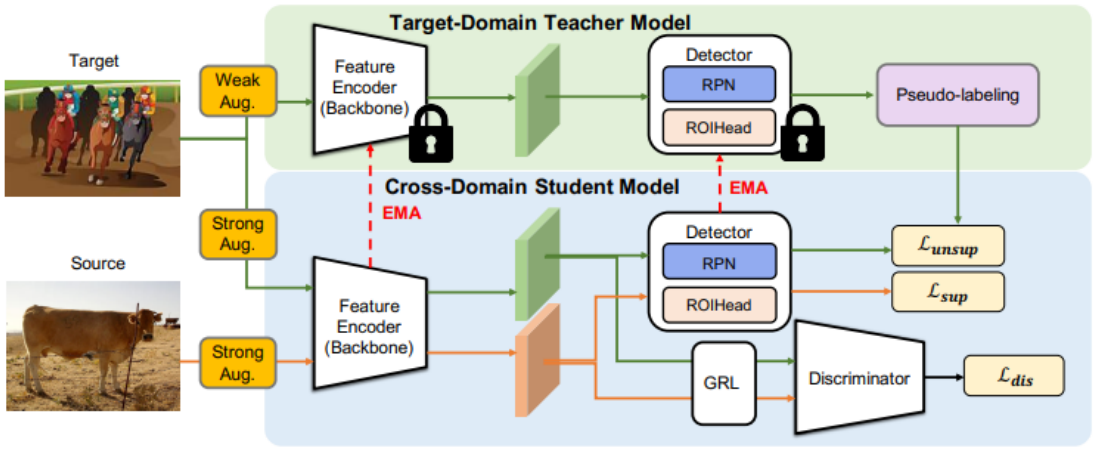


Figure 32: Cross-Domain Adaptive Teacher for Object Detection [11]

Liu et al.[70] 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 used in the adaptive teacher method.

Although all of the methods discussed in this chapter are based on the two-stage object detector Faster-RCNN, fewer papers also addressed a single-stage object detection in a domain adaptation setup, such as UDA for YOLO [71], [72], UDA for FCOS [73] and UDA for DETR [54], [74]. However, their review will be omitted

with the grounds for it given in the Research Methodology section.

2.7 Continual learning

While transfer learning attempts 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 [14]. 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" [14]. 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 33.

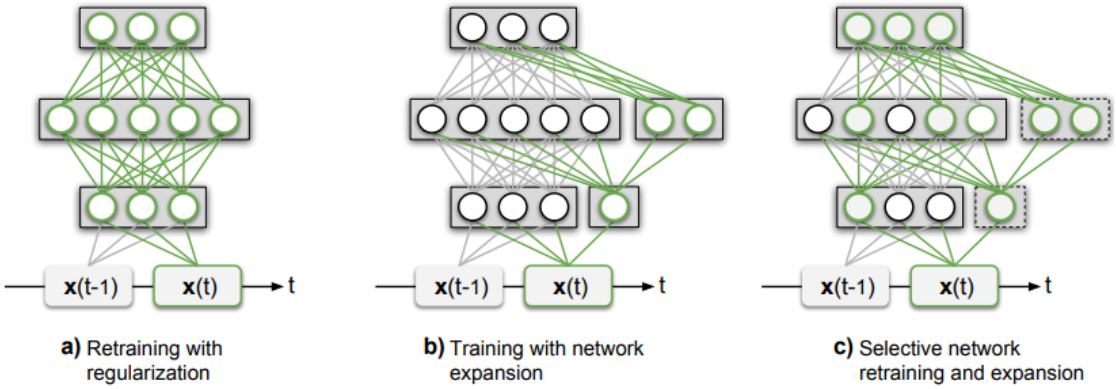


Figure 33: Continual learning approaches [14]

The methods proposed in Figure 33(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 method was presented by Razavian et al. [75], where the gradients calculation for the parameters of the original is disabled completely. Slightly modified version of this method was also proposed by Donahue et al. [76], where the learning rate was decreased to minimize the parameter update instead of blocking it completely.

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 proposed by Rusu et al. [77], where the network was trained on the initial task. As N new tasks were added, N number of layers were appended to the network, as shown in Figure 33(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. [78] falls into the third category, which is illustrated in Figure 33(c). The concept is fundamentally similar to the one suggested by Rusu et al.[77], 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. have collected many other promising contunual learning methods and the reader is advised to refer to the original paper.

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 $\mathcal{D}_n = \mathcal{D}_{n-1}$.

perhaps add the final CL graph from Parisi2018

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. Finally, the section will conclude with

3.1 Dataset

Initially, one of the objectives of this thesis was to implement an equipment identification system using images of the installed equipment and images of the 3D models to expand the dataset. Images of the 3D equipment parts could potentially be the main source of the dataset and they could be obtained from the 3D model of the gold refining plant of the customer of Metso Outotec. The entire plant's model is stored in Autodesk Navisworks in `.nwd` format and contains thousands of different equipment parts. It was discovered that Navisworks offers a possibility of rendering both images and videos of the desired equipment. However, 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 [79], a simple script with an extra toolbox were added Navisworks to filter out the required model, render and export its images in `.jpg` format. The results are presented in Figure 34 below.

Figure 34: Example of the rendered image of an arbitrary model

Upon a quick research, it was discovered that 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, unlike the dataset with rendered images, a dataset with images of real equipment turned out to be troublesome due to confidentiality and accessibility issues. Moreover, considering that rendered images would require further processing and labeling, it was decided to switch to an open-source dataset for the purpose of this project.

Previously, in the Neural networks in computer vision section, few datasets, such as ImageNet[30], PASCAL VOC[31] and COCO [32] 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) [10] meets the requirements. It consists of nearly 39 000 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. Unfortunately, the dataset is originally meant for 6D-pose estimation [10] as a part of the Benchmark for Pose Estimation(BOP) challenge [80]. As a result, the format of the dataset is derived from the format defined by BOP [81]. Therefore, a script was developed to convert it into a format commonly used by the state-of-the-art 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 [43] 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 setup is illustrated in Figure 35.

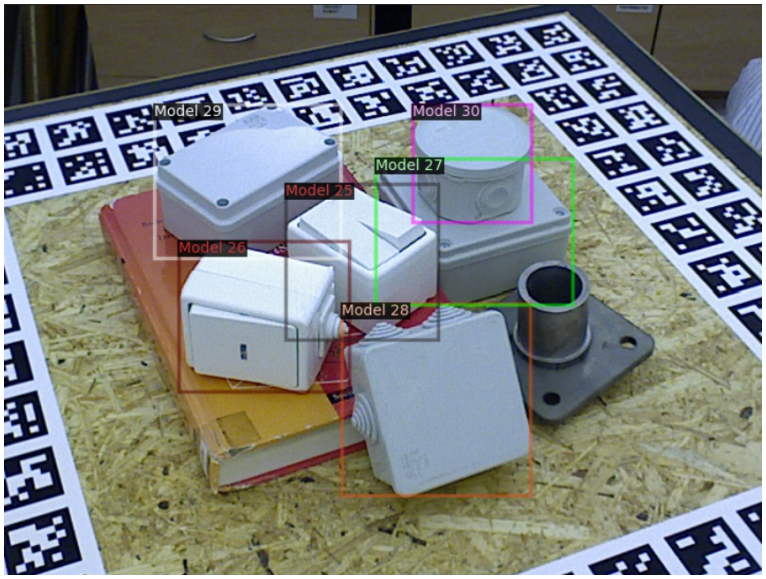


Figure 35: T-LESS real setup, labelled [10]

The names of the object classes presented in T-LESS, although undisclosed, are irrelevant to this thesis. Therefore, here and in the subsequent sections, when referred to individually, the classes will be named as "Model N". Additionally, in the experiments, the datasets with images from the simulator and the images from the real environment setup will be mainly referred to as "rendered" and "real" datasets, respectively. The images from the rendered dataset are saved in `.jpg` format, while the real images are stored as `.png` files, similarly as in the original T-LESS [10].

Furthermore, for this work, the rendered dataset will be split in 85%/15% proportion into the training and testing subsets. The distribution of the classes in the rendered dataset is presented in Figure 37.

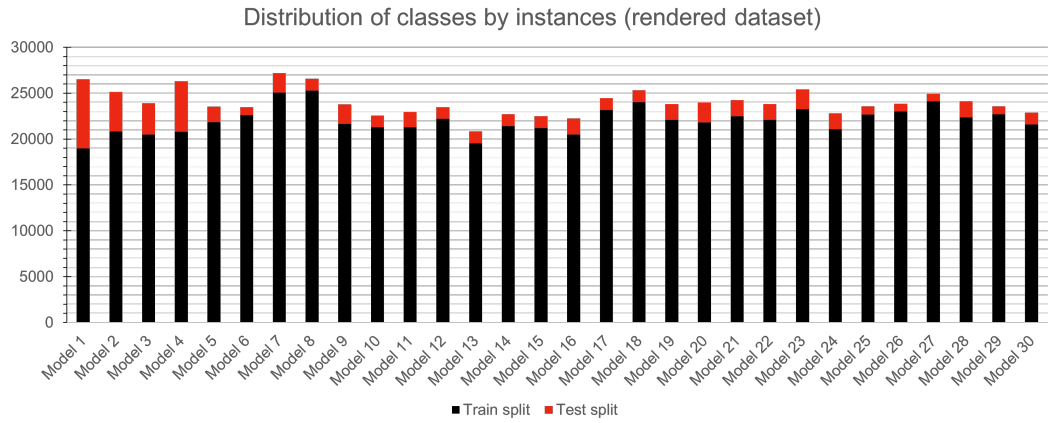


Figure 36: Distribution of the classes in the rendered subset of T-LESS dataset. 42 500 training images and 7 500 testing images. Total number of object instances: 720443

On the countrary, real images are only used for validation and their distribution by class is shown in Figure 37. The reasons for this will be discussed in later sections.

Talk about this distribution in future work

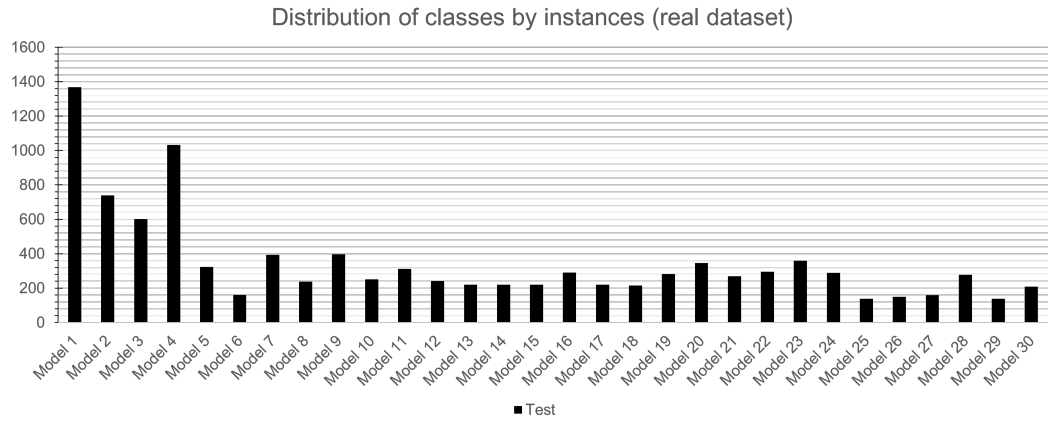


Figure 37: Distribution of the classes in the real subset of T-LESS dataset. Total number of object instances: 10362

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 [45] model was tested.

Additionally, YOLOv3 [52] was considered. However, unlike Faster-RCNN, YOLO has not been as popular, according to Figure 17. 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 metric(mAP) of the PASCAL VOC [31] challenge. In order to measure the mAP, first the confusion matrix should be calculated. Confusion matrix is a generic performance measurement tool in ML. According to PASCAL VOC, precision and recall are the most important terms in object detection tasks that can be extracted from the confusion matrix and their equations are shown in Figure 38.

		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** $\frac{TP+TN}{TP+TN+FP+FN}$

Figure 38: Definition of confusion matrix and some of its terms, adapted from [82]

To calculate precision and recall, it is important to define how the True Positive(TP) term is calculated. This is essentially done by measuring the Intersection-over-Union(IoU) metric. As the model proposes an anchor box, the classifier ouputs a confidence score. The confidence score is a probability of the box to contain an

object. The region is then additionally compared against the ground truth bounding box labels using the Equation 6. Finally, if the IoU ratio, also known as Jaccard distance, 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 [82] and only predictions with the highest confidence score count.

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 lower than the threshold, the value is labeled as False Negative(FN). Next, precision and recall values are calculated as specified in Figure 38.

$$\text{IoU}\left(\text{ Area }_{\text{Prediction}}, \text{ Area }_{\text{Ground truth}}\right) = \frac{\left|\text{ Area }_{\text{Prediction}} \cap \text{ Area }_{\text{Ground truth}}\right|}{\left|\text{ Area }_{\text{Prediction}} \cup \text{ Area }_{\text{Ground truth}}\right|} [82] \tag{6}$$

The area under the Precision-to-Recall curve is known as Average Precision(AP). The pairs of precision and recall are recorded at multiple confidence scores and compared against each other in the Precision-to-Recall curve. In PASCAL VOC 2010-2012[31], the area under the curve is smoothed first, and then the rectangles below the interpolated curve are summed as shown in Figure 39. The interpolation p_{interp} of the curve p is performed using Equation 7, where r is a recall level.

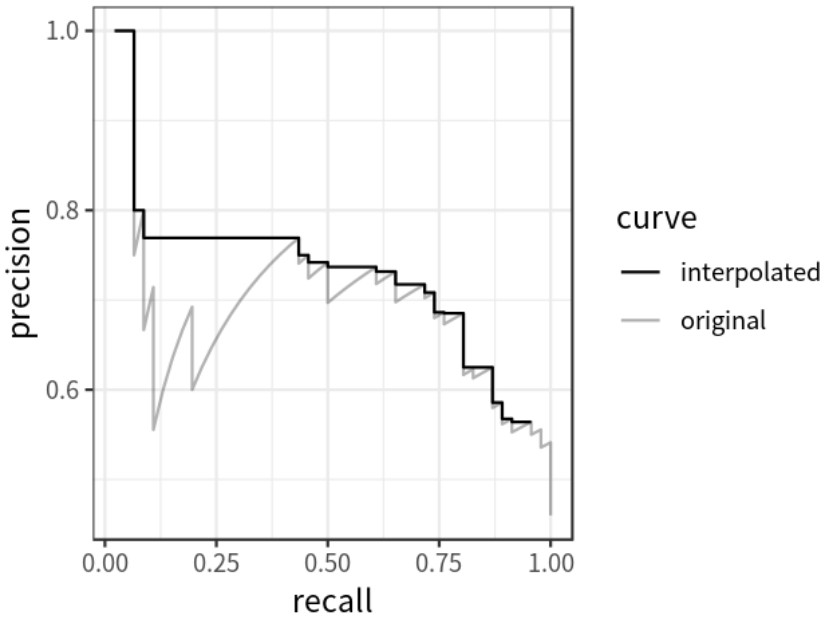


Figure 39: Average precision curve [82]

$$p_{\text{interp}}(r) = \max_{r' \geq r} p(r') [82] \tag{7}$$

In the subsequent sections, AP50 and AP75 denote average precision at IoU of 0.5 and 0.75, respectively, as defined by COCO [32]. According to PASCAL VOC [31] evaluation methods, AP stands for average precision at all confidence intervals, unlike COCO [32] and earlier iterations of PASCAL VOC, where AP was only calculated at 11 equally spaced fixed intervals.

3.2.2 Naive approach

After directly applying Faster-RCNN detector[12] in detectron2 [13] framework to the prepared dataset and evaluating it using the metrics presented earlier, it quickly became evident that the solution has to be more complex in order to solve the challenges outlined in the Thesis objective.

For the initial experiments, a Faster-RCNN network with a ResNet backbone was used. The ResNet architecture had 50 layers. The training has been conducted in three stages. First, the model was trained on the rendered dataset. Next, the trained model was evaluated on the real images of the same classes. Finally, identical model has been trained on the real dataset for comparison. After training the model for a short duration, corresponding to 20 000 iterations and the base learning rate of 0.00125, the following results were obtained:

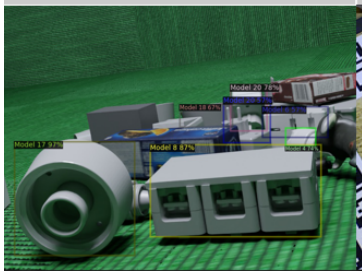
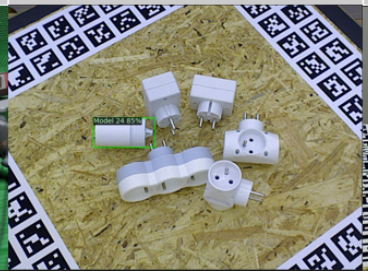
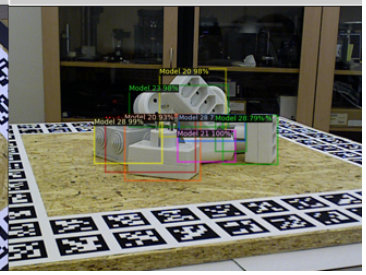
Model trained on rendered data and tested on rendered images	Model trained on rendered data and tested on real images	Model trained on real data and tested on real images
AP - 64.3510 AP50 - 76.0020 AP75 - 72.4191	AP - 9.0097 AP50 - 13.6304 AP75 - 10.046	AP - 83.3187 AP50 - 98.1990 AP75 - 94.2122
		

Table 2: Experiments with a simple Faster-RCNN model.

As it can be concluded, there is a significant performance drop when the environment is slighly changed. This is essentially a result of two related problems combined: overfitting and domain shift. Naturally, two datasets are originated from different environments and not only their background, but also their lighting conditions, positioning and textures deviate from one another.

3.2.3 Experiments with existing domain adaptive methods

To overcome the domain shift phenomenon, it was proposed to experiment with domain adaptation applications. 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 [83] 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 subproblems to avoid them from interfering with each other. According to the authors, this could improve the discriminability of the detector. Readers can refer to the original paper [83] for more information. The results of the experiments are outlined in the Table 3.

	Test on source data	Test on external (real data)
Model trained with only rendered data	AP 59.39	AP 6.77
Model trained on rendered data and adapted to real	AP 57.69	AP 36.85

Table 3: Results of the experiments with a D-Adapt based method.

First, a model was trained on the rendered dataset for 4 hours. Similarly to the Naive approach, the performance drops dramatically when testing the model on the real dataset. However, after running the adaptation network for another 4 hours, the results have improved by a considerable margin. The final result on the combined dataset is $AP = \frac{57.69+36.85}{2} = 47.27$. Although the result is much better than the one without any adaptation with $AP = 9.0097$, this performance seems to be rather insufficient.

Recently, many open-source solutions proposed [84] [85] [86] an Image-to-Image translation approach in a cross-domain object detection setup. Following Zhu et al. [67], Cycle-GAN 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 4.

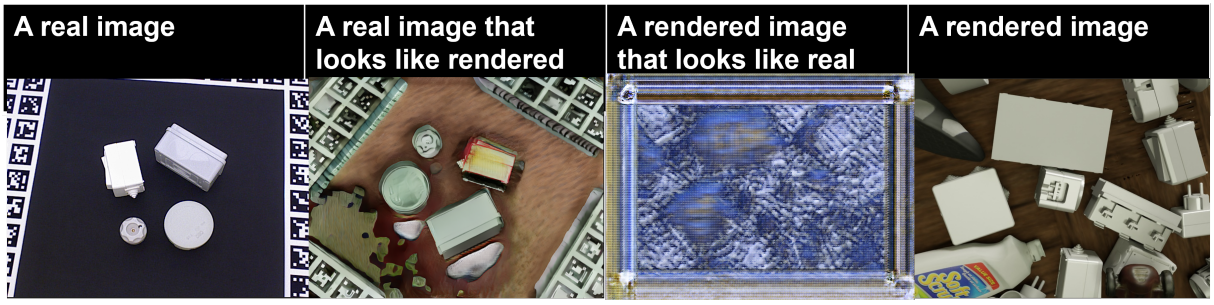


Table 4: Results of the experiments with Cycle-GAN

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 limitations of Cycle-GAN [67], where differences in the distribution characteristics of the training dataset caused comparable artifacts. Due to the low performance on the T-LESS dataset and long training time, this method was excluded from further tests.

3.3 Implementation details

For the final set of experiments, an ensembled setup was proposed. 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 in the Mean Teacher and Graph Reasoning section.

In this thesis, the adaptive teacher framework is studied extensivelyas

4 Validation and Results

TODO

5 Analysis and Discussion

TODO

6 Conclusion and Future Work

TODO

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

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

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

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

[8] TempAuthor, "Tempcitation."

[9] 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.

[10] 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.

[11] 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.

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

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

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

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

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

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

[19] 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.

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

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

[22] 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.

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

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

[25] 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.

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

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

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

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

[39] 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.

[40] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[55] 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.

[56] 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.

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

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

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

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

[61] 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.

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

[63] 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.

[64] 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.

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

[66] 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.

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

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

[73] 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.

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

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

[76] 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.

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

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

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

[80] 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.

[81] 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.

[82] 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.

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

[84] 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.

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

[86] 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.

A Appendices

TODO