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** 92+2 **Language** English

Abstract

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

This thesis attempts to minimize the manual labeling process by proposing a regularized cross-domain adaptive teacher model with continual learning. The model assumes a task that seeks to eliminate the domain shift between industrial datasets: a larger labeled dataset of rendered images and a smaller unlabeled dataset of real-life images. While the labels for the rendered images can be generated automatically, only a tiny amount of real images needs to be collected, which is crucial for the system scalability in industrial environments.

The model transfers knowledge from one domain to another by means of adversarial domain adaptation and mean teacher training. In an attempt to achieve state-of-the-art results, this thesis proposes to regularize the student and the teacher networks using image-level alignment, instance-level alignment and consistency loss. Additionally, the model adopts a lifelong learning approach with network expansion and gradient regularization that enables the model to be retrainable on a continuously expanding dataset, which further facilitates the scalablity of the system.

As a result, the proposed Adaptive teacher model with two-level alignment and additional augmentations achieved competitive results with max (AP50) = 69.57 % at 14 999 iterations, which is twice as fast compared to the original model with max (AP50) = 71.39 % at 30 999 iterations. On the other hand, the continual learning experiment with 5 arbitrary classes proved that retraining the model on the entire dataset (max (AP50) = 69.66 %) brings more benefit than training the model continuously using the proposed approach (max (AP50) = 43.87 %).

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

Preface

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

Otaniemi, 29.7.2022

Saidnassimov, D.

Contents

List of Figures

List of Tables

Symbols and abbreviations

Symbols

λ	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

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

List of Abbreviations

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

Todo list

2 Background

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

2.1 Deep learning and neural networks

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

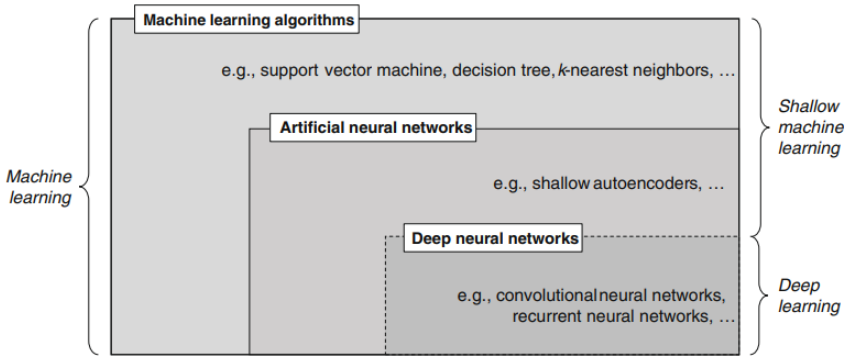


Figure 1: Machine learning concepts [26].

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

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

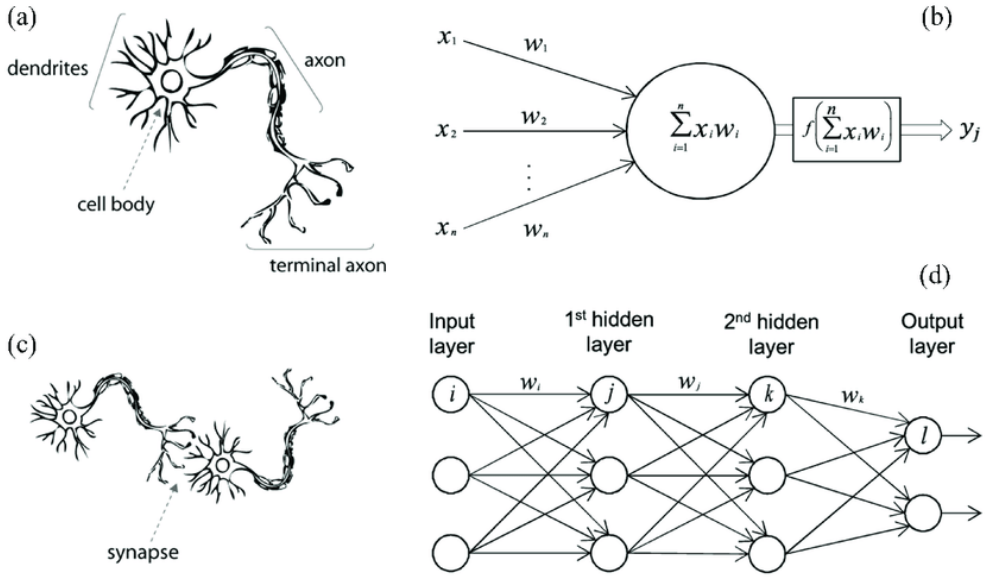


Figure 2: A biological(a) neuron against artificial(b) and biological synapse(c) against artificial(d) [28].

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

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

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

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

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

Due to the biological nature, neural networks adapt over time by creating new connections between neurons. The neurons in ANN adopted such behaviour by utilizing a backpropogation algorithm [31]. A naive backpropogation approach is illustrated in Figure 3. The algorithm consists of two parts: feed-forward and backward loops. Generally, the main objective of an ANN is to choose such weights that the network produces desired target outputs. The forward pass propagates along the nodes 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. The equation to MSE is shown in Equation 2.

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

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

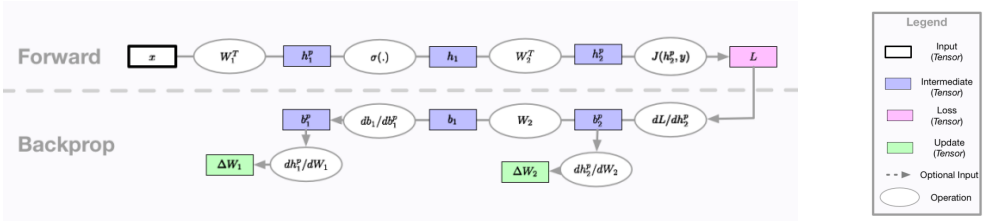


Figure 3: Backpropogation algorithm, adapted from [33].

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

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

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

2.2 Neural networks in computer vision

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

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

For addressing the key objectives of this work (Section 1.2), the thesis will extensively cover image classification, object detection and domain adaptation tasks. Figure 4 illustrates a simplistic CNN architecture approach to the MNIST [36] classification problem. A CNN is essentially a neural network that leverages convolutional layers to produce predictions. Unlike the traditional computer vision methods, CNNs do not need to extract features of the image beforehand due to the logic behind convolution. In classical ML the features are extracted separately, followed by the appropriate algorithms for learning. On the contrary, DL algorithms, such as CNN, learn the features automatically [37].

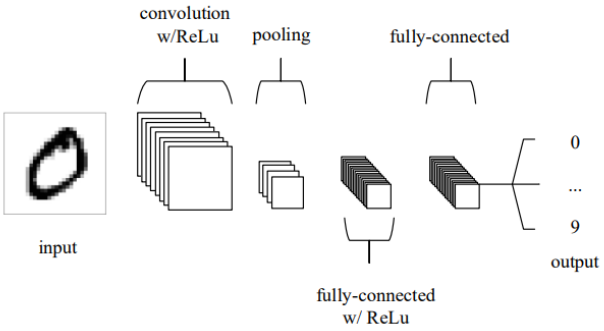


Figure 4: A simple CNN network with 5 layers for image classification task [34].

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

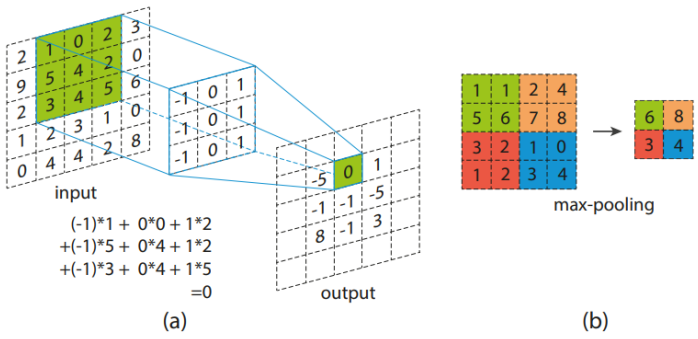


Figure 5: The process of convolution [38].

As a result, the elements of the network are not densely connected, which allows better generalization and flexibility. This operation in practice allows to extract the important features of the image, such as edges, corners, shapes and many others. However, unlike in the FC layers, the number of weights is much smaller, which is essential when it comes to high-dimensional images. Following the architecture

in Figure 4, the outputs of the convolutional layers are activated with ReLU in a similar manner as in a classical ANN structure. The outputs are then pooled in order to down-sample the image and hence reduce the computational costs [38]. The commonly used max-pooling operation is shown in Figure 5 (b). Max-pooling down-samples the image by applying a filter that extracts the maximum value from the region of a feature map. Finally, fully-connected layers complete the structure of the basic CNN. FC layers are used to flatten the outputs from the convolutional layers. In case of the input in Figure 4, this in turn allowed to compute the probability of the input image to represent a number between 0 and 9 [34].

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

2.3 Image classification

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

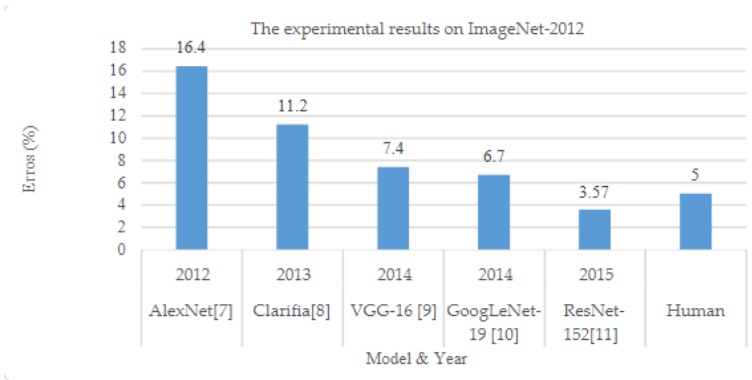


Figure 6: Evolution of image classifier models evaluated on ImageNet dataset [37].

2.3.1 LeNet

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

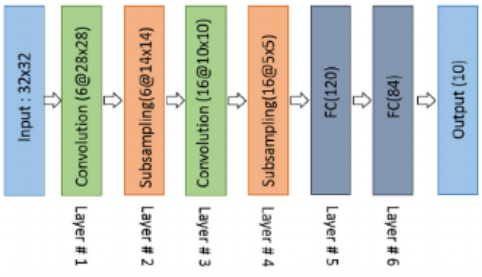


Figure 7: LeNet architecture [37].

2.3.2 AlexNet

Consequently, AlexNet paper was introduced, which proved the effectiveness of their model, as it outperformed the state-of-the-art implementations and achieved the error rate of 15.3% [43]. Figure 8 displays the proposed network. The architecture of AlexNet is similar to one of LeNet, though it is substantially deeper and has more than 60 million adjustable parameters. It has 5 convolutional layers of varying kernel size. The convolutional layers are 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 over-fitting in FC layers and improve generalization [46].

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

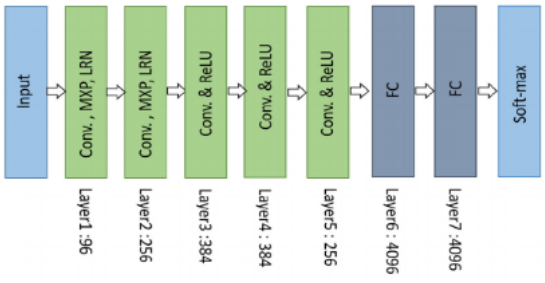


Figure 8: AlexNet architecture [37].

2.3.3 VGG

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

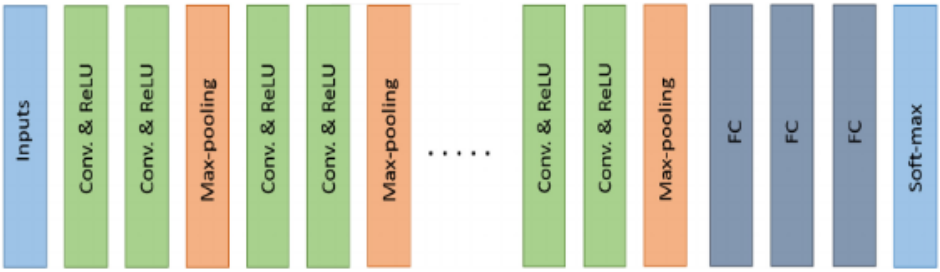


Figure 9: VGG architecture [37].

2.3.4 ResNet

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

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

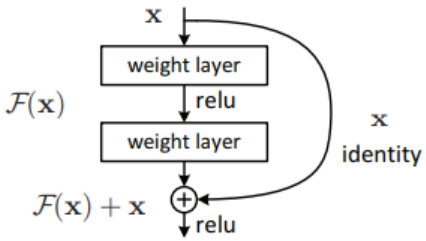


Figure 10: Residual block of ResNet [45].

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

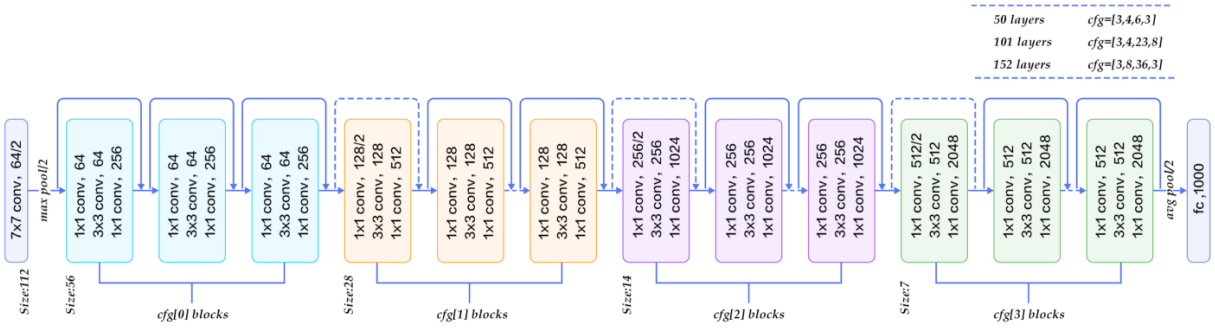


Figure 11: ResNet architecture [47].

2.4 Object detection

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 pre-trained 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 [48], as shown in Figure 12.

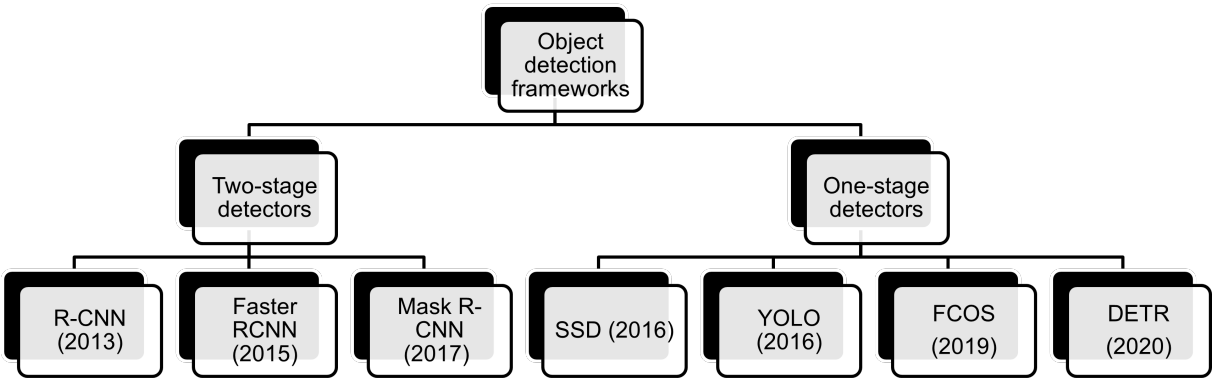


Figure 12: Types of object detectors.

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

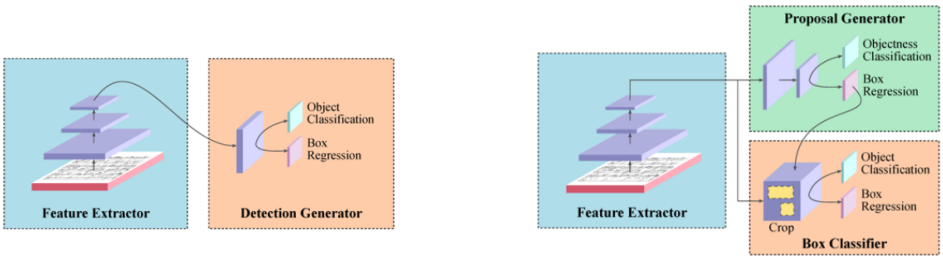


Figure 13: A simple single-stage detector(left) compared to a two-stage detector(right) [49].

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

2.4.1 R-CNN

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

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

2.4.2 Fast-RCNN

Unlike the classic R-CNN system, where the components such as the classifier and the regressor have to be trained separately, the same authors of the RCNN model also introduced a Fast-RCNN model [52], which proposed an end-to-end trainable system. Additionally, the region-of-interest(ROI) pooling layer was proposed. This ROI component essentially divides the image into a grid of a fixed size and applies average pooling. The resulted one-level pyramid of the image is used as a feature map for the detector head. This allows the rest of the network to focus purely on

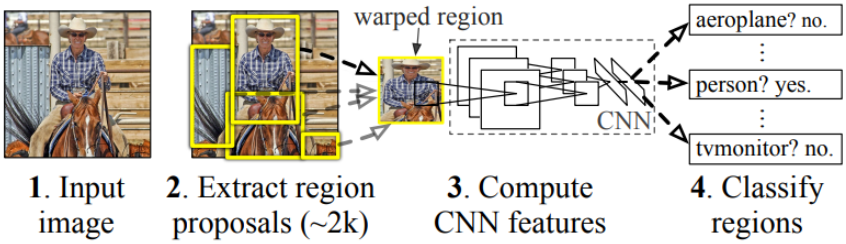


Figure 14: R-CNN overview [51].

the features extracted for the proposed regions. Finally, two additional FC output layers are appended, where the first one classifies the features by returning the probability of N object classes in range $(0, N + 1]$, which includes the background class as well. The other one returns a vector of a size $(0, 4 \times N]$, thus including the bounding box coordinates for the foreground classes only [52]. Fast-RCNN showed a significant($\times$146) speed improvement as compared to the R-CNN, thus allowing it to be used in real-time applications [51]. The original architecture of Fast-RCNN is shown in Figure 15 below.

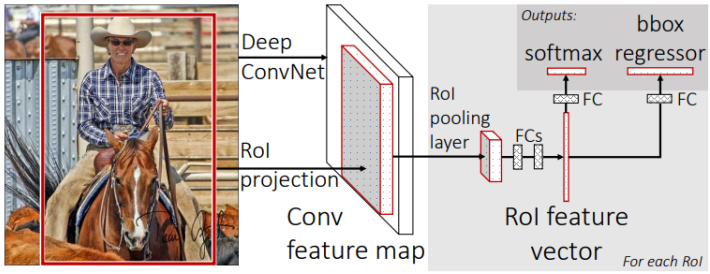


Figure 15: Fast-RCNN overview [52].

2.4.3 Faster-RCNN

Ren et al. [12] discovered that even though Fast-RCNN was significantly faster than the preceding networks, the region proposing process was still causing a bottleneck. Therefore, in the novel Faster-RCNN [12] architecture it was suggested to replace the legacy region proposal module with a fully-convoluted architecture called region proposal network(RPN) [51]. Instead of using the one-level image pyramid to solve the problem as it was proposed by Girshick et al. [52], Faster-RCNN utilizes anchor

boxes of different aspect ratios to propose object candidates. Similarly to any other CNN, the features in Faster-RCNN are extracted from the convolutional layers based on a VGG backbone. The features maps are then sent to the RPN, which in practice is a sliding window of $n \times n$ $(n = 3)$ dimensions. At every sliding window position, the k number of object proposals are predicted. Such boxes are called "anchors" as illustrated in Figure 16. The acquired $2k$ classification predictions whether the proposed region is an object of interest or not and 4 regression outputs of the bounding box coordinates are then mapped back to the ROI layer and, eventually, to the FC layers that predict the class of the proposed object [12].

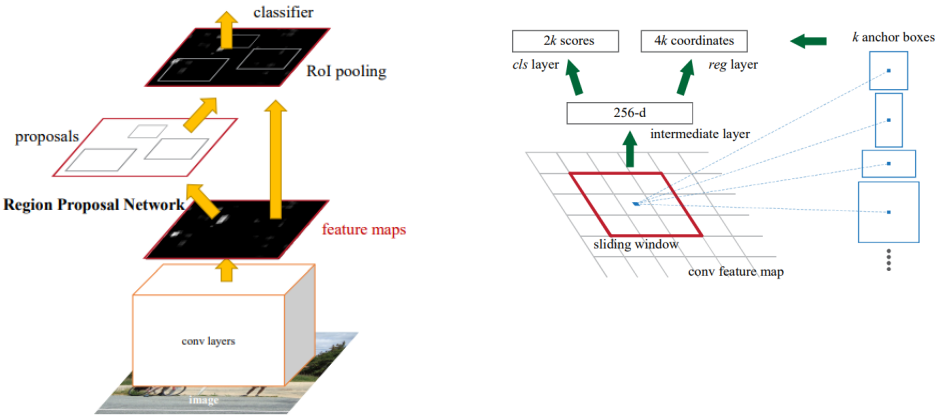


Figure 16: Faster-RCNN overview and its RPN module [12].

As the network returns the predictions, the loss function is calculated as in Equation 3:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}} \sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{reg}\left(t_i, t_i^*\right) \text{ [12]}, \tag{3}$$

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

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

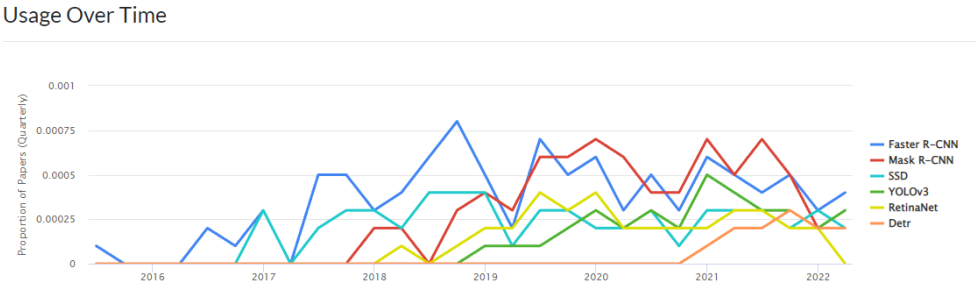


Figure 17: The popularity of different detectors according to [56].

2.4.4 YOLO

Unlike the two-stage methods presented, You Only Look Once(YOLO) [15] algorithm 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 bounding boxes and C probabilities of the class and are predicted [15]. The overview of the detection process is shown in Figure 18. The principle of the YOLO grid component is broadly similar the one of R-CNN [51], where the algorithm of selective search [54] is used to propose regions. However, instead of proposing more than 2000 regions, YOLO only returns 98 proposal boxes per image. This, along with having an optimized single-stage detection process, allowed YOLO to achieve impressive nearly real-time FPS results. Redmon et al. reported YOLO to sustain the average FPS of 45, while the most accurate version of the Faster-RCNN network had 7 FPS [15]. However, the proposed network still has multiple limitations, which were addressed in later iterations of the paper [57, 58].

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

2.4.5 SSD

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

However, as the objects might not always be within the grid boundaries, as can

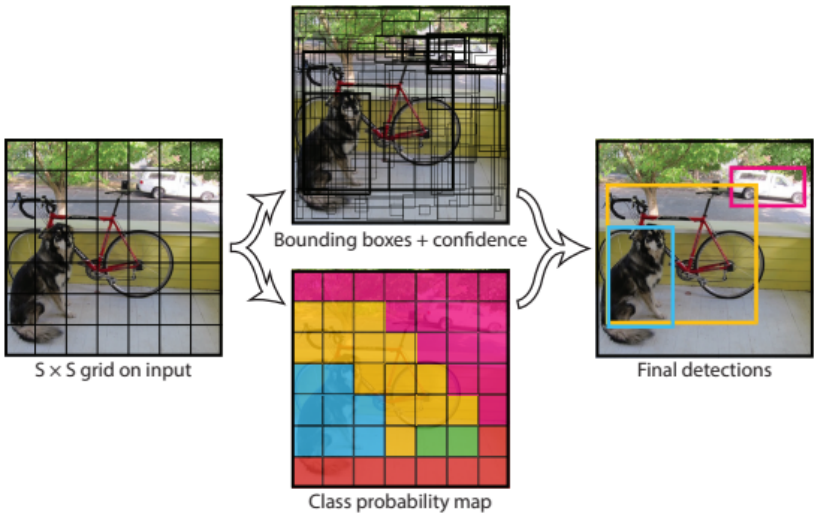


Figure 18: YOLO overview [15].

be noticed from Figure 20. Therefore, SSD paper introduced a concept of anchor boxes with offsets. The anchor boxes with offsets that have the highest overlap with the ground truth box of the object are then passed to FC layers to predict the class and the location of the object.

The key object detection models of the last decade are presented in 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 of Zaidi et al. [48] also discusses lightweight models. However, for the purpose of this study, the detection accuracy is the primary focus rather than speed, thus their review will be omitted.

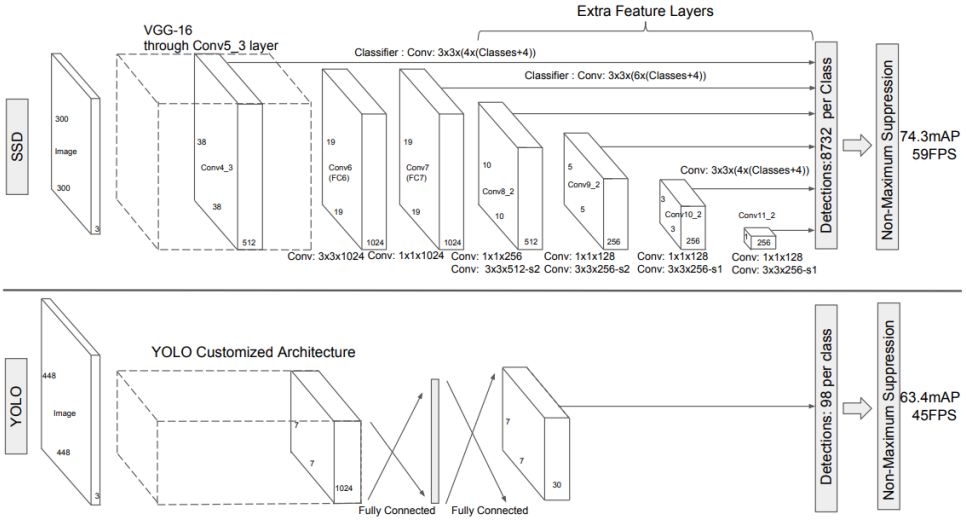
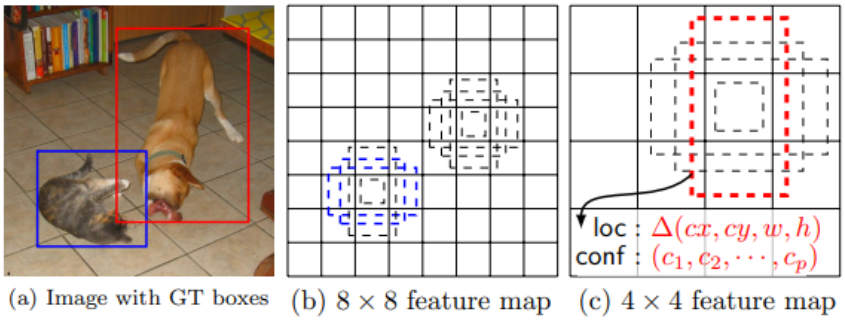


Figure 19: SSD compared to YOLO [13].



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

Figure 20: SSD anchor boxes [13].

Table 1: Overview of object detectors [48].

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

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

As it can be seen from Table 1, there are multiple models that are not covered in this section. Additionally, there are several detectors that show promising results in the domain-adaptive object detection setup, such as FCOS [17] and DETR [16]. However, as their performance is not yet extensively evaluated, they will be omitted from this thesis.

2.5 Object detection in industrial environments

Although object detection is an extremely diverse field with applications varying from medicine [2] to self-driving vehicles [3, 4], the industrial object detection is mostly limited to safety and process monitoring [1, 5]. Fewer research has addressed the topic of industrial object detection. In general, many of the introduced object detection methods can be applied to this problem directly as it is. Malburg et al. [7] has evaluated multiple different versions of object detection models such as YOLO [15] in an industrial setup. However, often in industrial environments severe challenges such as overlapping and occlusion take place. Moreover, the object instances often are represented in multiple scales. Therefore, it is often impossible to obtain reliable results using a standard object detector architecture. Wu et al. [6] has recently addressed the problem of industrial object detection using an R-CNN based approach that leverages Feature Pyramid Networks to improve detection at multiple scales and NMS with penalty factors to eliminate occlusion problem. However, the proposed method does not consider the data availability limitations and confidentiality issues that were outlined in the Thesis objective. On the other hand, Kim et al. [8] minimized the need for labeling from a human by utilizing a deep active learning model. While the core of this approach is a standard Faster-RCNN [12] network, Kim et al. also proposed to utilize uncertainty evaluation in order to enable the model to label images interactively over time. As the training advances, the model evaluates whether the predicted bounding boxes are reliable or not using the sum of entropy for each bounding box. In the subsequent stage the model requests the user to manually label 10% of the data with highest uncertainty. The model is then retrained with the updated knowledge. Although this approach seems promising, it does not consider the domain shift phenomenon that takes place in the setup defined by the Thesis objective. Later sections of the thesis will address this phenomenon.

Write more about this active learning method if there is time

2.6 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 in terms of time, financially, and sometimes simply not possible at all due to data accessibility issues [59]. Moreover, the requirement of having to train the models on new massive datasets often make ML solutions inefficient in practice. For this reason, TL concepts have been found advantageous as 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. [60] 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 $\mathcal{D}_\mathcal{S}$ and one target $\mathcal{D}_\mathcal{T}$ domains [60].
- 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 [60].

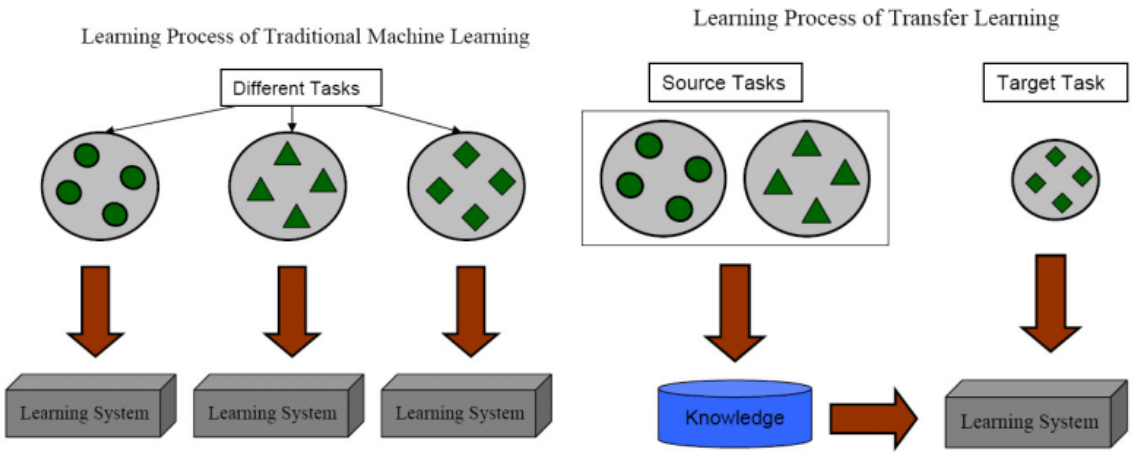


Figure 21: Comparison of ML to TL [60].

Sun et al. [61] 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. [60] mentions that transfer learning can be also defined when the marginal probability is different, meaning that $P_S(X) \neq P_T(X)$. Similarly, transfer learning can be applied when $\mathcal{T}_S \neq \mathcal{T}_T$. An example of a simple TL problem is training a model that was trained for classifying cats, but instead is required to learn a new task such as classifying dogs. However, in various scenarios, the task is essentially the same but the domain is similar yet different. For instance, a model that was trained to identify cats in sketches, is instead required to identify cats in real images. Solving such tasks is the main focus of domain adaptation(DA). Often the terms "domain adaptation" and "transfer learning" are used interchangeably. However,

according to Wang et al. [62] and Zhang et al. [63], 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 $P(\mathcal{X}_S) \neq P(\mathcal{X}_T)$ [61]. This is not to be confused with semi-supervised machine learning, where both labeled and unlabeled data is typically supplied from one domain [61].

2.6.1 Domain adaptation

Domain Adaptation (DA) has lately been gaining popularity in both image classification and object detection tasks [63]. 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 [63]. In a DA problem, a domain shift, also known as a distributional shift or dataset bias, can be defined as a change in distribution of data between source and target domains.

Zhang et al. [63] classify DA methods into three categories that are similar to ML types - supervised, semi-supervised, and unsupervised DA. Alternatively, Oza et al. [64] classify the collected DA methods into semi-supervised, weakly-supervised and unsupervised DA. Indeed, supervised DA is not commonly used since the primary goal of DA is to reduce the domain shift when the data availability is limited.

According to Zhang et al. [63], 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 in Figure 22 and the methods in the following subsections attempt to minimize the domain shift by one way or another.

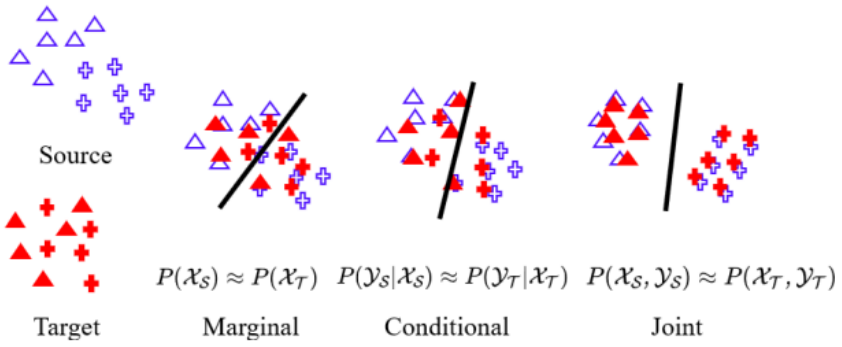


Figure 22: Distribution alignment types [63].

2.7 Domain adaptive object detection

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

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

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

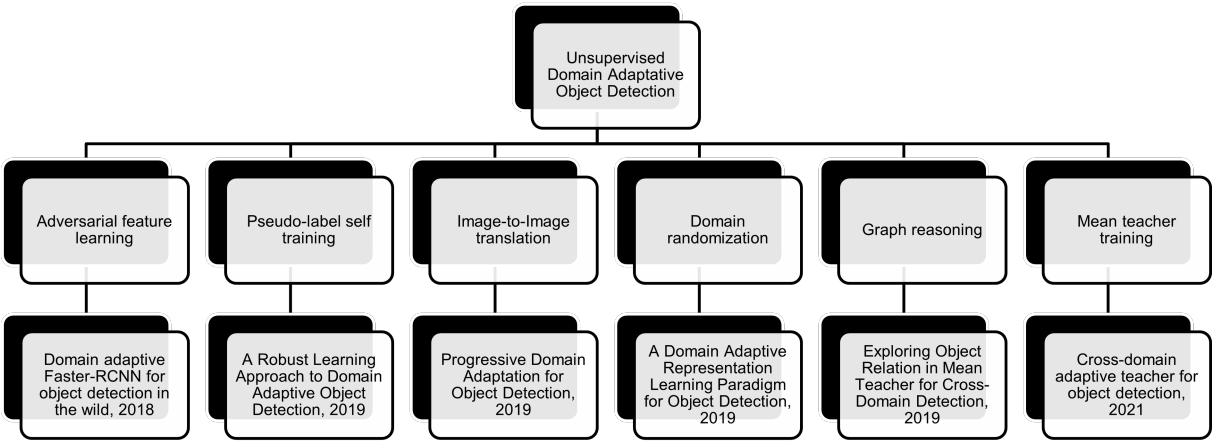


Figure 23: Unsupervised Domain Adaptive Object Detection.

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 address unsupervised domain adaptation(UDA) problem, which is naturally more sophisticated than a supervised DA.

2.7.1 Gradient reversal layer

One of the key components in a typical domain adaptive setup for both image classification and object detection problems is a gradient reversal layer(GRL) that has been proposed by Ganin et al. [18]. 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. Figure 24 illustrates the Domain-Adversarial Neural Network (DANN) proposed by Ganin et al. [18].

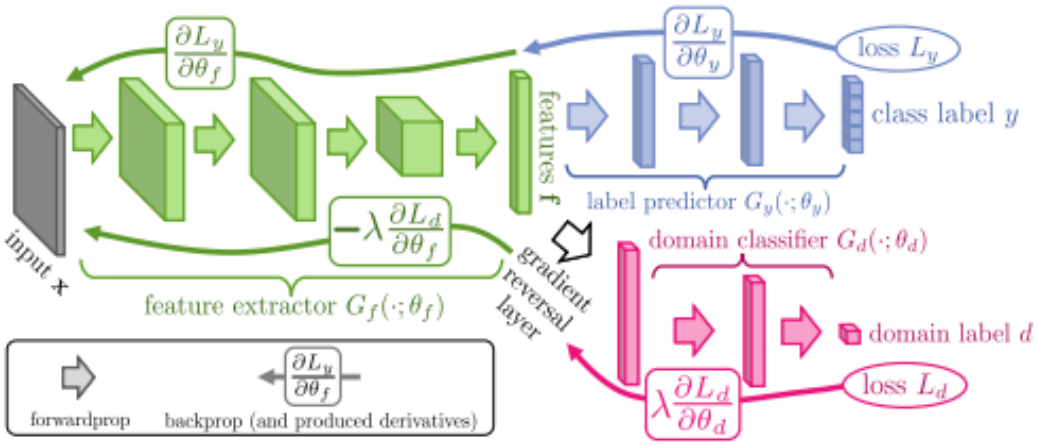


Figure 24: Domain-adversarial neural network and GRL [18].

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 [18].

Perhaps add more math about GRL

2.7.2 Adversarial feature learning

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

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

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

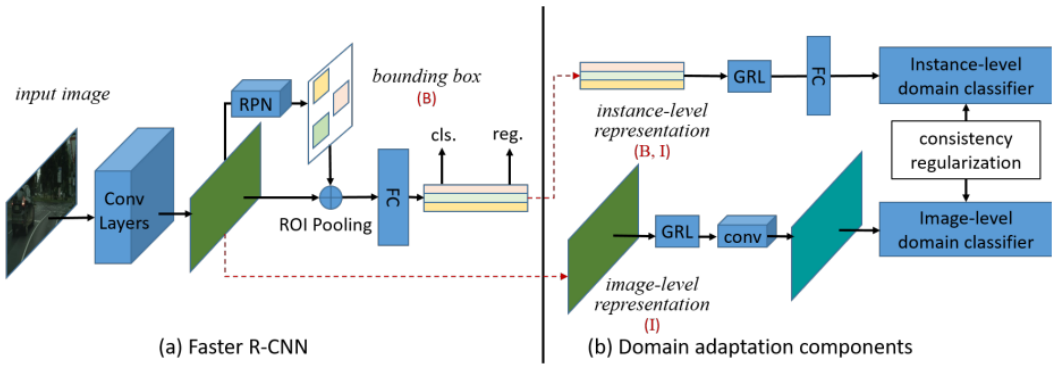


Figure 25: Domain Adaptive Faster R-CNN for Object Detection in the Wild [22].

Here, to maximize the domain classification losses L_{inst} and L_{img}, they are calculated as shown in Equation 4.

$$\begin{aligned} \mathcal{L}_{img} &= -\sum_{i,u,v}\left[D_i \log p_i^{(u,v)} + (1-D_i)\log\left(1-p_i^{(u,v)}\right)\right] \\ \mathcal{L}_{ins} &= -\sum_{i,j}\left[D_i \log p_{i,j} + (1-D_i)\log\left(1-p_{i,j}\right)\right] \end{aligned} \quad [22], \tag{4}$$

where $D_i = 0$ is a ground-truth label of the sample image i that originates from the source domain and $D_i = 1$ originates from the target domain; $p_{i,j}$ is a prediction output of the instance-level classifier for j-th proposed region of the sample image i; $p_i^{(u,v)}$ is a prediction output of the image-level domain classifier for the feature

map region (u, v) of a sample image i. Finally, two domain classifiers are regularized using consistency loss $\lambda\mathcal{L}_{\text{consistency}}$ as shown in Equation 5.

$$\mathcal{L}_{\text{consistency}} = \sum_{i,j} \left\| \frac{1}{|N|} \sum_{u,v} p_i^{(u,v)} - p_{i,j} \right\|_2 \text{[22]}, \tag{5}$$

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

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

Another most recent state-of-the-art study made by Rezaeianaran et al. [67] proposed different approach that compares the adversarial training to contrastive learning. Similarly to Chen et al. [22], the network leverages a Faster-RCNN detector with instance- and image-level alignments. Rezaeianaran et al. attempted a different approach by pushing the features closer if they represent the same class, and push them apart if otherwise. This is performed by utilizing a max-margin contrastive loss. The margin here denotes how far the features can be in order to be considered to represent the same class. A general constrastive loss term takes the form of Equation 6:

$$\mathcal{L}_{CL} = \sum_{i}^{C} \left[\left\| F_S^i - F_T^i \right\|_2^2 + \sum_{j,j\neq i}^{C} \max\left\{ 0, m - \left\| F_S^i - F_T^j \right\|_2^2 \right\} \right] \text{[67]} \tag{6}$$

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

This paper review should be rewritten

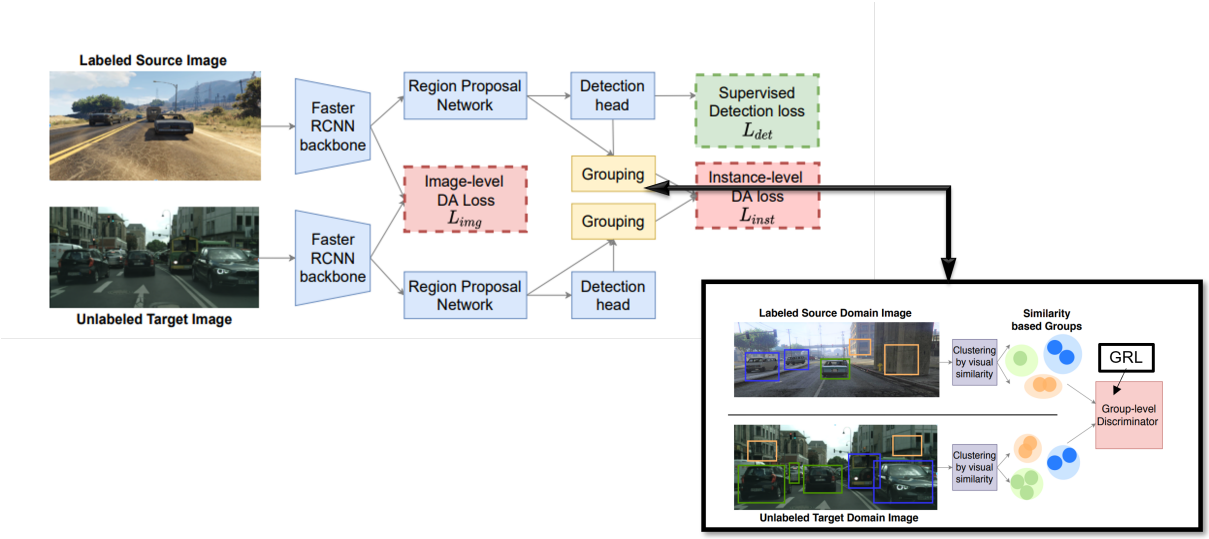


Figure 26: Seeking Similarities over Differences: Similarity-based Domain Alignment for Adaptive Object Detection, adapted from [67].

2.7.3 Pseudo-labeling based methods

Another relatively straightforward method to solve an object detection problem can be done by using pseudo-labels. A naive approach to pseudo-labeling is to first train the source dataset $\mathcal{D}_{\mathcal{S}} = \{\mathcal{X}_{\mathcal{S}}^i, \mathcal{Y}_{\mathcal{S}}^i\}_{i=1}^{\mathcal{N}_{\mathcal{S}}}$ 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 $\dot{\mathcal{D}}_{\mathcal{T}} = \left\{\mathcal{X}_{\mathcal{T}}^j, \mathcal{Y}_{\mathcal{T}}^j\right\}_{j=1}^{\mathcal{N}_{\mathcal{T}}}$ [64]. However, the results obtained will naturally be noisy and of poor quality. Khodabandeh et al. proposed a three-phase training process illustrated in Figure 27 below.

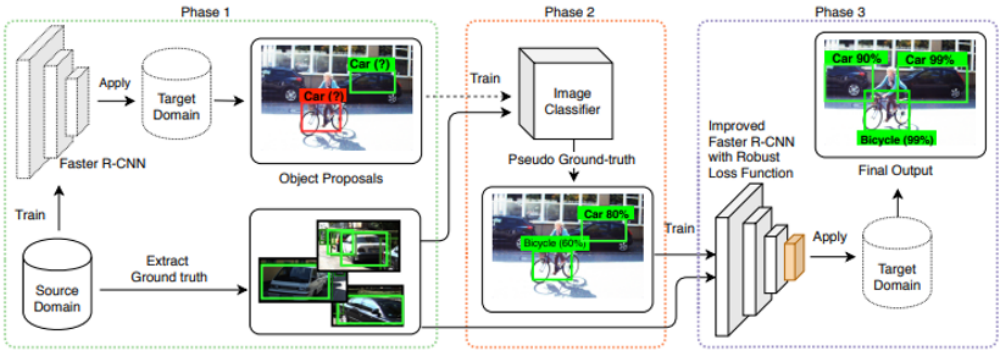


Figure 27: A Robust Learning Approach to Domain Adaptive Object Detection [68].

In the 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 pre-trained image classifier, which allows to refine the model.

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

$$\min_{q} \mathrm{KL}\left(q\left(y_c\right) \| p_{cls}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right) + \alpha\, \mathrm{KL}\left(q\left(y_c\right) || p_{img}\left(y_c \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_l\right)\right) [68], \tag{7}$$

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 [68].

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

2.7.4 Image-to-Image translation

Another category that Oza et al. [64] outlined is image-to-image translation for UDA. Instead of trying to align features, this group of methods essentially attempt to pull the domains together first. Hsu et al. [69] has suggested a Cycle-Generative Adversarial Network(GAN) [70] 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.

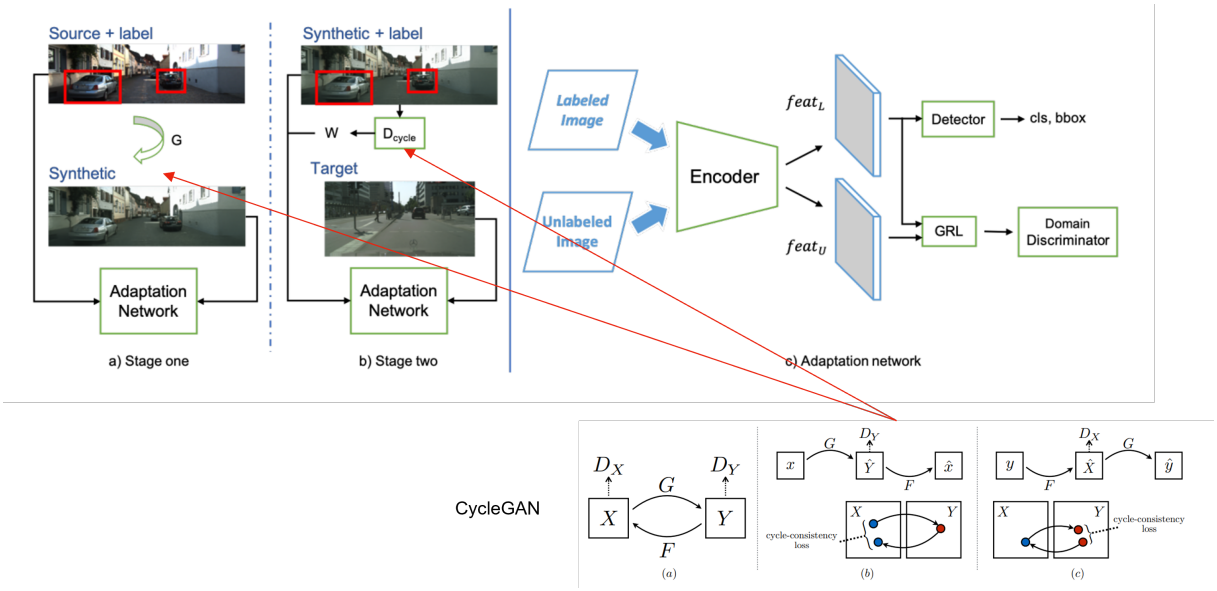


Figure 28: Progressive Domain Adaptation for Object Detection and CycleGAN, adapted from [69].

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 synthetic domain. To further minimize the domain shift, adversarial learning techniques are used, such as a combination of a GRL and a domain classifier [69].

In Figure 28 (b), the synthetic source-alike images are then passed together with the inherited source labels back to the adaptation network to align 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 [69].

2.7.5 Domain randomization

Oza et al. [64] argues that often the accuracy of image-to-image translation methods is questionable as the domain shift between the synthetic and source domains still exists. Slightly different approach has been offered by Kim et al. [71]. Instead of trying to pull two given domain distributions closer, they proposed a domain randomization technique, which generally attempts to generate a brand new domain

that includes the same image in different style. This in practice allows the detector network to recognize features that are domain invariant and remove the domain bias.

Kim et al. 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. [69], the module leverages a Cycle-GAN [70]. 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.

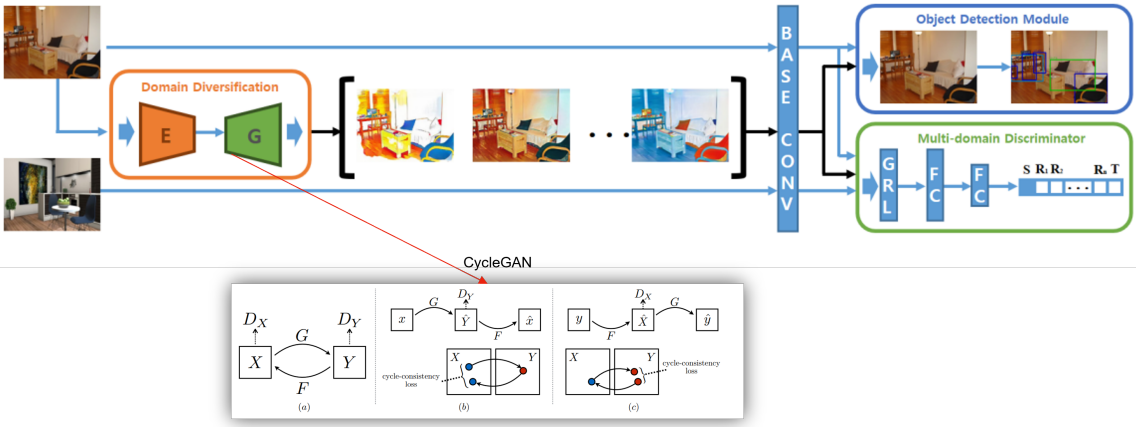


Figure 29: Diversify and Match: A Domain Adaptive Representation Learning Paradigm for Object Detection, adapted from [71].

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 [64]. In the original experiments Kim et al. considered three additional domains: a color preserved domain, a reconstructed domain, and a domain that combines both. Images from all the domains are then fed into the detector along with the inherited source labels to train a domain-invariant network with help of a multi-class domain classifier and the resulted model was used to verify performance on the target dataset.

2.7.6 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 [64]. Cai et al. [72] proposed an architecture that combines both mean teacher and graph reasoning techniques in one solution and such architecture is presented in Figure 30.

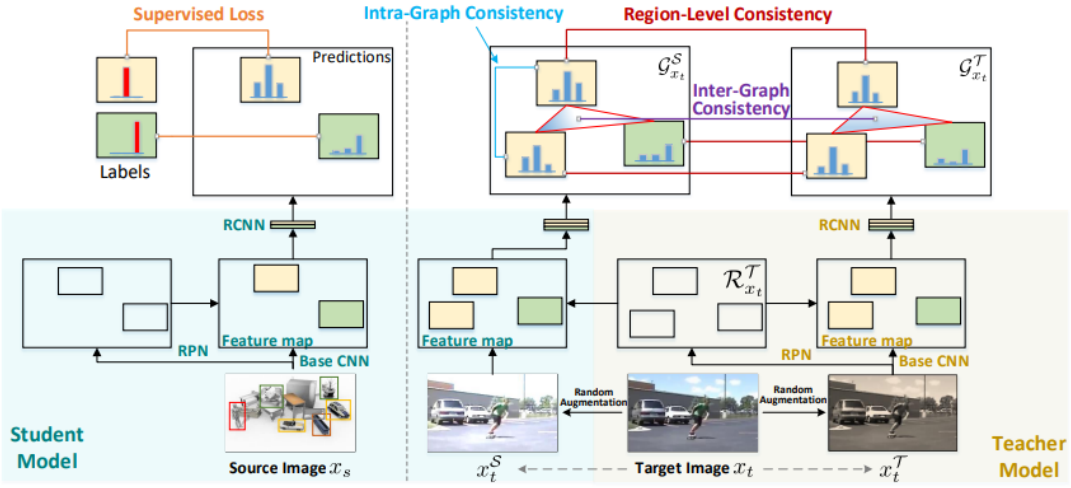


Figure 30: Exploring Object Relation in Mean Teacher for Cross-Domain Detection [72].

The term "graphs" represents a complex data structure that contains multiple nodes or vertices connected to each other via edges. In case of an image as a graph, each pixel can be considered a node, which is linked to all of its neighbors. The setup developed by Cai et al. [72] introduces a student-teacher framework based on Faster-RCNN that verifies the 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. On the other hand, images from the target domain are augmented in two steps. First, the images are randomly cropped, padded or flipped. These images are passed through the pre-trained supervised student model that generates predictions. Meanwhile, the original target images are augmented with color jittering or corruption with noise. This set is send to the teacher model. The teacher model also produces predictions, which are then compared with the predictions from the first set of augmented images by utilizing region-level consistency. Essentially, the region-level consistency is calculated as MSE of the region-level prediction error in both the student and the teacher.

Inter-graph level consistency is used to verify the quality of the two graphs 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 the UDA setup, the closest prediction argmin(*labels*) is used to produce pseudo-labels. The intra-level consistency loss is then calculated for any two instances of the same class in one graph. For more detailed calculation of the loss terms the readers are referred to the original paper by Cai et al. [72].

A purely mean-teacher-based semi-supervised approach has been proposed by Liu et al. [73] 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.

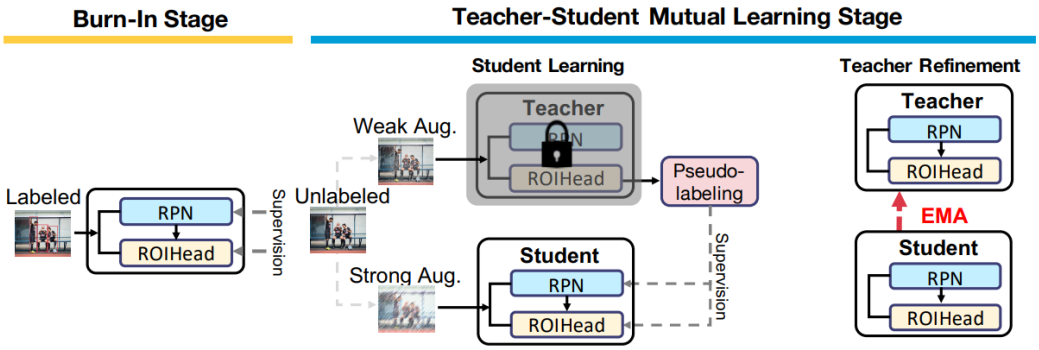


Figure 31: Unbiased Teacher for Semi-Supervised Object Detection [73].

In order to achieve higher accuracy of pseudo-labels, the unbiased teacher model includes an Exponential Moving Average(EMA) module, along with a few other techniques [73]. EMA attempts to emphasize the most recent data by granting it a higher weight. The EMA in Unbiased teacher is defined as follows:

$$\theta_t^i = \hat{\theta} - \gamma \sum_{j=1}^{i-1} \left(1 - \alpha^{-j+(i-1)}\right) \frac{\partial \left(\mathcal{L}_{\text{sup}} + \boldsymbol{\lambda}_u \mathcal{L}_{unsup}\right)}{\partial \theta_s^j} [73], \tag{8}$$

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(LR) of the ensembled model [73].

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

However, this method only solves a semi-supervised object detection without addressing the domain shift issue. Expanding the unbiased teacher method, Li et al. [20] proposed to utilize mean teacher training in a domain adaptation setup. In addition to the original ensembled network proposed by Liu et al. [73], adaptive teacher architecture employs adversarial feature learning techniques to 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.

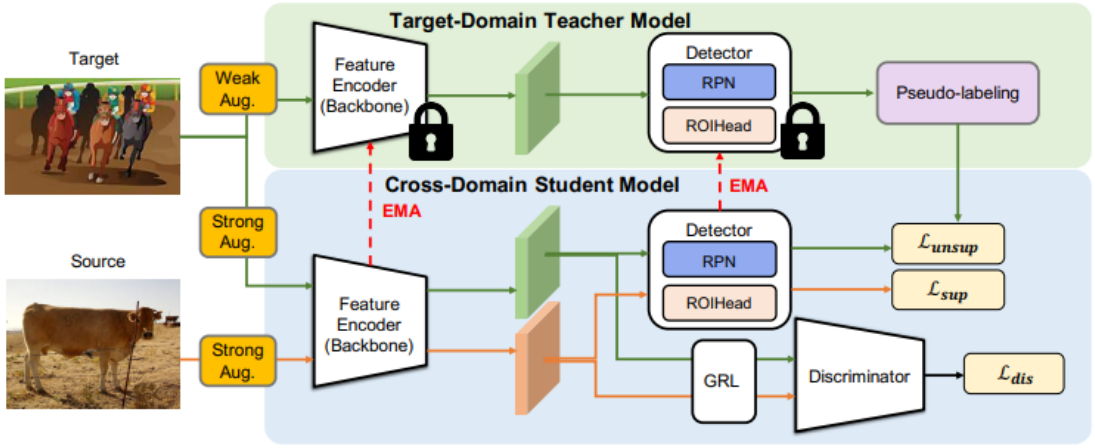


Figure 32: Cross-Domain Adaptive Teacher for Object Detection [20].

Liu et al. [73] specified that the teacher model is supplied with weakly-augmented target images, while the student model uses both strongly-augmented source and target images. Weak augmentations include cropping and flipping the image horizontally, and strong augmentations included grayscaling, color jittering, Gaussian blurring and cutting out patches. The same strategy was employed in this adaptive teacher method. The augmentations used in these works [20, 73] are presented in Figure 33.

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 [75], [76], UDA

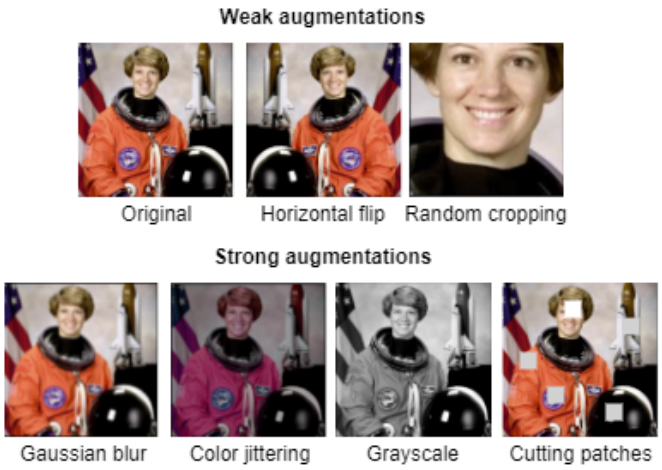


Figure 33: Augmentations used in Unbiased teacher (adapted from the official Pytorch documentation [74]).

for FCOS [77] and UDA for DETR [16], [78]. However, their review will be omitted with the grounds for it given in the Research Methodology section.

2.8 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 [23]. In case of the simple object detection problem, continual learning can be applied to a new task, such as to learn new classes of objects that are supplied progressively over time. This approach is potentially useful due to the scalability benefits it brings, since retraining the entire model every time a new object arrives to the database is computationally expensive.

Similarly to the human brain, ANNs in object detectors tend to forget old knowledge learned as the memory gets overwritten with fresh data. This phenomenon in continual learning is commonly addressed as "catastrophic forgetting" [23]. Parisi et al. summarizes the up-to-date methods of continual learning that are effective against catastrophic forgetting into three categories: retraining with regularization, selective training with network expansion and retraining selective network with expansion. These methods are illustrated in Figure 34.

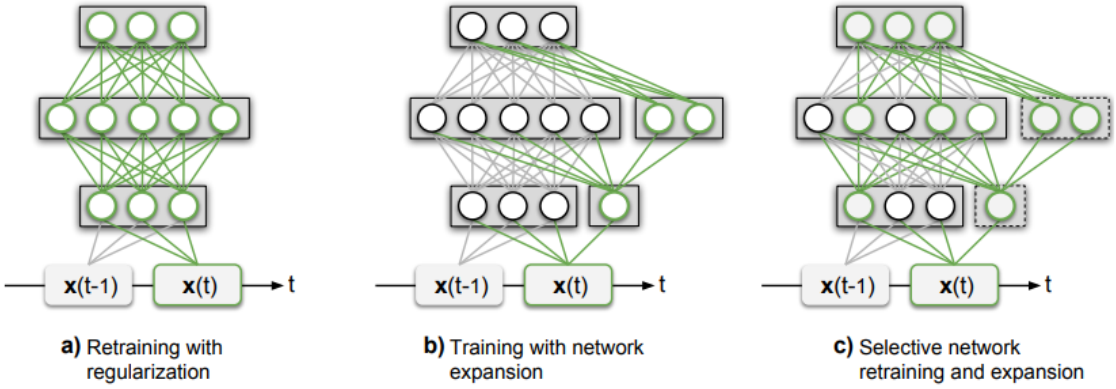


Figure 34: Continual learning approaches [23].

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

In simplistic terms, the method of training with expansion denotes retraning the original network with an additional number of layers appended. A typical example of such a method was proposed by Rusu et al. [81], 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 34 (b). The parameters of the original network were left unchanged, and only the additional connections were trained. Although the disadvantage of this method was that the network complexity would grow linearly over time as new tasks are added, the experiments delivered sufficient performance.

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

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

A Appendices

A.1 Architecture

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

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

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